Learning Patterns from Imbalanced Evolving Data Streams

Manal Almuammar
School of Computer Science and Electronic Engineering
University of Essex
Colchester, UK
Email: manalm@essex.ac.uk

Maria Fasli
Institute for Analytics and Data Science
School of Computer Science and Electronic Engineering
University of Essex
Colchester, UK
Email: mfasli@essex.ac.uk

***Abstract*—Learning patterns from evolving data streams is challenging due to the characteristics of such streams: being continuous, unbounded and high speed data of non-stationary nature, which must be processed on the fly, using minimal computational resources. An additional challenge is imposed by the imbalanced data streams in many real-world applications, this difficulty becomes more prominent in multi-class learning tasks. This paper investigates the multi-class imbalance problem in non-stationary streams and develops a method to exploit real-time stream data and capture the dynamic of patterns from heterogeneous streams. In particular, we seek to extend concept drift adaptation techniques into imbalanced classes' scenarios, and accordingly, we use an adaptive learner to classify multiple streams over a sequence of titled time windows. We include examples of the falsely classified instances in the training set, then we propose using a dynamic support threshold to discover the frequent patterns in these streams. We conduct an experiment on the car parking lots environment of a typical University with three simulated streams from sensors, smart pay stations and a mobile application. The result indicates the efficiency of applying adaptive learner approaches and modifying the training set to cope with the concept drift in multi-class imbalance scenarios, it also shows the merit of using a dynamic threshold to detect the rare patterns from evolving streams.**

***Index Terms*—data stream; pattern discovery; imbalanced classes; evolving stream; rare pattern**

I. Introduction

The rapid evolution of technology has moved our world toward a new era of intelligence. The remarkable development in sensors and communications has led to the emergence of the Internet of Things (IoT); a network of physical devices, vehicles and other objects which are embedded with sensors, software and network connectivity that enable these objects to interact with their surrounding environment, collect and exchange data [1]. According to the report of Cisco [2], 3.3 billion devices are expected to be connected to the Internet by 2021. Indeed, the data which flow from a large numbers of sensors to report on environmental conditions or deployed in infrastructure (such as roads and buildings), can give decision makers a heightened awareness of real-time events.

The current trend toward data streams has introduced new challenges. Learning from data streams is quite different to learning from traditional data [3]: not only in the short processing time and the computational resource constraints when dealing with streaming data, but also in that the traditional techniques assume stationary environments whereas in most real-life applications (e.g., weather predictions, financial markets or monitoring systems) the environment is non-stationary and data are evolving over time in a phenomenon called concept drift [4] [5]. Therefore, while all available data are considered in the traditional techniques, streaming algorithms restrict processing to a certain window of concern, focusing on the most recent elements in the stream.

Another common problem in real-world applications is class imbalance [6]. Typically, it occurs in classification scenarios where the classes are not represented equally in the data set. In these cases, some of the classes may be rare, or appear only occasionally in the data stream, like in fraud detection, spam filtering or fault diagnosis in computer monitoring systems [7] [8]. In such cases, the classifiers tend to be biased toward the majority classes, resulting in a high accuracy classification performance, however these classifiers will not be able to detect any instance of the minority class. Multi-class imbalance imposes additional challenges compared to two-class ones, the situation may be exacerbated in the data stream domain as the data are dynamically evolving and it is impossible to see the whole picture of data [6] [7] [9]. In general, re-sampling is the simplest and most effective imbalance technique for both static and streaming data [7].

In the field of data mining, patterns discovery has become a powerful tool for extracting valuable information from the massive amount of data. Apriori [10] was the first proposed algorithm in the literature to mining frequent patterns, it was designed for market basket analysis. Subsequently, many other algorithms were proposed without candidate generation such as the FP-growth algorithm [4], but the rare pattern problem was a major issue in these works, as they used a single threshold to identify the frequent patterns over the entire data set. To solve the rare patterns problem, multiple minimum supports were introduced by Liu et al. in [11]. Up to now, several works have been proposed in the literature, such as Conditional Frequent Pattern-Growth and CFP-Growth++ [12], however, the construction cost of the conditional subtrees restricts extending them in streaming scenarios.

Motivated by the increasing proliferation of the IoT technologies in our daily life, where heterogeneous data streams flow from multiple sources, this work investigates the com-

bined challenges posed by evolving streams and multi-class imbalance, while it is trying to capture the dynamic patterns in these streams. The complexity of the problem lies in the intersection of stream processing, concept drift and imbalanced data. The combination of these difficulties constitute a challenge not only when building an algorithm to learn from non-stationary streams but also when evaluating its performance. To our knowledge, the ensemble techniques approach is the only method for overcoming the concept drift which has been applied to the class imbalance problem in the literature [11], more research needs to be undertaken to extend other concept drift adaptation techniques (adaptive base learners and learners which modify the training set) in the context of class imbalance.

In this paper, we propose a Frequent Pattern learning method from Evolving Streams (FP-EStream). We extend concept drift adaptation techniques into imbalanced classes scenarios by developing an adaptive learning algorithm using a windows based approach. We also modify the training set by keeping a window of the minority classes to cope with the imbalanced classes problem. To find interesting patterns from the incoming data streams, we use a dynamic support threshold instead of the fixed one in the FP-growth algorithm.

The rest of this paper is structured as follows. The following section provides the required background and the related work. Then our proposed algorithm is demonstrated and discussed in Section III. In Section IV, the experimental work follows; the description and the simulation of the dataset, building the classification model and discovering the frequent patterns. Finally, the conclusions and the future directions of our research are covered in Section V.

II. Background

A. Stream Reasoning

A data stream is defined as an unbounded and ordered sequence implicitly by arrival time or explicitly by time stamp of data items [13], while stream reasoning is the analysis of these data in real time to extract hidden information and support decision making. There are a variety of streaming data scenarios, it could be a continued flow of data items that arrive in timely order or it might be a huge amount of data which possibly have infinite length. Another common scenario is where data objects' values are changing frequently at high speed like the stock exchange, or where the data stream is generated continuously by multiple data sources, which typically send in the data records simultaneously, and in small sizes, like video or sensors streams.

B. Concept Drift

One of the biggest challenges to classify data streams in the real-world is concept drift; this phenomenon occurs when the data streams are generated in non-stationary environments. For example, weather prediction models change seasonally, and customer behaviour changes over time according to the fashion and the season [14] [15]. The change of data streams distributions over time may lead to the emergence of new classes or changes to an existing class [16]. In fact, concept drift can cause a degraded predictive performance of the classifiers over time, therefore, the learning model must detect the change and adapt itself to the current state of the environment [17] [18]. Previous studies have distinguished between two types of drifts [15][16][19]:

- Real concept drift which indicates the changes in the conditional distribution of the target variable given a set of input variables, regardless the change in the inputs' disruptions.
- Virtual drift occurs when the distribution of the incoming data changes $p(x)$ without affecting the conditional distributions of the target variables $p(y \mid x)$.

Hence, in both cases the joint distribution $p(x, y)$ changes.

Several adaptive techniques have been proposed in the literature to detect and cope with the drift in an evolving data stream. Change detectors, such as the Drift Detection Method [20] and Adaptive Sliding Window Algorithm [21] are standalone techniques that detect concept drift and can be used in combination with any stream classifier [22]. The authors in [15] attempted to put adaptive techniques works into three groups: window based approaches, weight based approaches, and ensemble classifiers [15]. In general, ensemble approaches are considered popular methods for stream classification due to their robustness [19].

C. Imbalanced Classes Distribution

Several works have been developed in the literature to tackle imbalanced classes in static data, most approaches aim to achieve a more balanced distribution by using various forms of re-sampling [23], however, similar research in the context of data streams is limited to few works [6], and too little attention has been paid to the multi-class imbalance issue [7]. In brief, imbalance data streaming approaches can be categorized into data level and algorithm level approaches [7]. One of the recent works to cope with the multi-class imbalance in data streams was presented in [24], the authors used a time decay function to detect the imbalance rate dynamically and accordingly they proposed two re-sampling-based ensemble methods, oversampling-based Online Bagging (OOB) and undersampling-based Online Bagging (UOB). A more recent technique was proposed in [25], this depends on an ensemble method which consists of voting based classifiers, and different classes weights are used to detect the imbalanced class in the validation set. However, this technique requires an initialization step before the learning, and it is not clear how the class weights will be accurate in the case of evolving streams.

Class imbalance introduces challenges to classifier performance measures and evaluation procedures used in stream mining. Many performance measures have been introduced to replace the overoptimistic accuracy, the most popular performance evaluation strategies are the prequential test [33], G-mean, and Kappa statistic [3] [9]. Also the use of F-score and area under the Receiver Operating Characteristic curve (ROC

curve) are common, but they are not suitable for multi-class tasks [17] [23] [9].

D. Mining Frequent Patterns

Frequent patterns are items that appear in the dataset with frequency under a user specified threshold. Frequent pattern mining was introduced in 1994, when Agrawal and Srikant proposed the Apriori algorithm [10], where association rules are simply statements that certain groups of items or events tend to occur together. There are two basic concepts related to association rules, the first is the confidence or accuracy which defines how often the rule is true. The second is the support or the coverage which indicates the frequency of a rule. An association rule is an implication of the form: $A \Rightarrow B$ where: $A \subseteq I, B \subseteq I$, and $A \cap B = \Phi$, then we can define: $Support(A \Rightarrow B) = p(A \cup B)$ and $Confidence(A \Rightarrow B) = p(B \mid A)$. Association rule mining consists of two steps, first finding all frequent item sets then generating rules from these sets. An item set is considered as a frequent one when its support is at least equal to the minimum support threshold [26].

Many other algorithms have been developed to detect different patterns in the transaction data. There are three basic frequent pattern mining methodologies: Apriori, FP-growth and Eclat [27]. FP-growth is distinguished from the other methodologies because there is no candidate generation, it works in a divide-and-conquer way and requires less number of database scans, so it is more suitable for huge database transactions and long patterns. It simply scans the database first to construct a list of frequent items in which items are ordered by frequency descending order, then according to the frequent list, the database is compressed into a frequent-pattern tree, or FP-tree, which retains the itemset association information. The FP-tree is mined to find frequent patterns instead of depending on the whole data set, by starting from each frequent length-1 pattern, constructing its conditional pattern base, then constructing its conditional FP-tree, and performing mining recursively on such a tree [4]. However, FP-growth suffers from a major limitation, as it uses a single support threshold to identify the frequent pattern for the entire dataset. In many real-world applications, choosing a low threshold value will identify a large number of meaningless patterns, whereas a high threshold value will lead to skipping rare patterns [28].

Many algorithms have been proposed to discover frequent patterns in data streams [27] [29], but most of them do not differentiate recently-generated information from obsolete information that may currently be useless, insignificant, or possibly invalid [30] [31]. A recent work in [27] developed the FP-Stream algorithm to mine frequent patterns in data streams. The algorithm is based on a batch environment, for each batch, the frequent patterns are extracted by means of the FP-growth algorithm applied on a FP-tree structure representing the sequences of the batch, however, due to the fixed support, the rare patterns are not detectable.

III. Learning Patterns from the Imbalanced Evolving Data Streams

In this section, we formulate the problem of learning patterns from the imbalanced evolving streams and then introduce our framework. We consider the problem in the area of a heterogeneous stream reasoning and where data coming from different sources, sensors and devices such as smart pay stations, mobile phones etc. can be used to extract useful information and provide support to users in a situation where there are multiple parking lots and different types of users that are seeking to park their vehicles.

Let S be the set of structured data streams $\{S_{t1}, S_{t2}, S_{t3}\}$ generated by different IoT devices at given time stamp T where:

$S_{t1}(f_1, \ldots, f_i)$	is a stream from camera sensors $sn_1 ... sn_m$; m: number of the sensors that monitor the entry of the set of parking lots
$S_{t2}(f_1, \ldots, f_k)$	is a stream from smart parking pay stations $ps_1 ... ps_l$; l: number of the parking pay stations
$S_{t3}(f_1, \ldots, f_z)$	is a stream from parking mobile application
i, k, z	are the number of fields in each stream

Let the first stream S_{t1} consist of the location of the sensor, crossing-time, car's plate number, and if the car's direction is in or out. The second stream S_{t2} (payment station stream) comprises the location of the station, car's plate number, parking-time, end of the parking duration, paid fee, and if a discount is applied. Finally, the third stream S_{t3} (mobile phone app stream) consists of the parking location, car's plate number, parking-time, end of the parking duration, paid fee, and the driver type from the user's profile. Let the joint of the three streams (based on the car plate number) in a fixed time interval (e.g., single hour) represent the current window-batch S_T, and the joint of the three streams in the previous time interval (the previous hour) represent the previous window-batch S_{T-1}. As the window-batches are time based windows, so their sizes vary, each window may consist of any number of data items in the joint stream. This is a commonplace scenario in car parking lots, although the structure of streams might be slightly different.

Mainly, we are not explicitly interested in identifying the frequent items from the various streams, but we are interested in developing methods to discover patterns which represent a meaningful situation in the car parking lots and which can be used to build predictive models, i.e. capture the behaviour of different drivers, so the driver will be able to know, before s/he comes to a location such as a university or place of work, how busy the parking lots are and the probability of finding a parking space when s/he arrives. This will lead to more efficient use of parking spaces.

Extracting the interesting patterns form evolving data streams is hard and time consuming as a huge number of patterns could emerge from them, therefore, the patterns should be identified according to the level of interest. For the purpose of this work, we propose the FP-EStream algorithm,

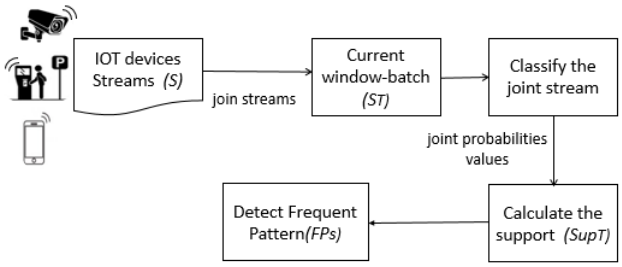

Fig. 1. Pattern Discovery Abstract Model

which is based on an integration of two mining techniques; classification (to extract the interesting patterns only) and frequent pattern mining (based on a modified version of FP-growth algorithm and using dynamic support). An abstract model of our approach for pattern discovery from evolving IoT data streams is shown in Fig. 1. Once we have received a window-batch of the joint data stream for the current time stamp (S_T), we first classify the stream then we calculate the dynamic support threshold value (Sup_T), next we use this threshold to discover the frequent patterns and finally we use the misclassified instances if any, to modify the upcoming training windows in order to identify the rare patterns.

Before we discuss the details of our proposed technique, and as the streams come from heterogeneous sources, it is important to note that streams must be pre-processed before applying our approach, this includes:

- Data cleaning, the data streams from sensors typically contain noise which affects the data analysis, so we have to fill the missing values, remove redundant ones and remove outliers.
- Data integration, joining the streams to process them in parallel over the same interval.
- Data transformation, data may have to be converted from one format or structure into another format or structure.
- Data reduction, to obtain the optimal minimum representation in volume which produces similar analytic results.

A. Titled Time Windows

As we mentioned above, we use time window-batches in our approach. More specifically, we will use different titled time windows in order to detect the periodical patterns (the daily and seasonal patterns) from the evolving streams over different time intervals. The titled time window is automatically self-maintained, whenever reaching the boundary of a time window granularity, the aggregates stored in a low granularity level are summarized and transformed to the upper granularity level (e.g. four windows of 15-minutes-window form one hour-window). Choosing the optimal length of the window depends on the application domain, in general, the window sizes should be adapted according to the drift speed in the data stream, so small windows should be chosen during periods of fast drift, while windows should be larger during periods of slow drift.

B. Classify the Imbalanced Evolving Streams

We choose naive Bayes (NB) as the base learning classifier, it is a supervised learning approach that assumes the independence between attributes and uses the probabilities of each attribute belonging to each class to make a prediction of another attribute class. In particular, it is based on the Bayesian theorem, which indicates that there is a simple relationship between $p(x \mid y)$ and $p(y \mid x)$, which can be expressed as:

$$p(x \mid y) = p(y \mid x) * p(x)$$

Let the previous defined joint stream S be represented by a vector $f = (f_1, \ldots, f_x)$ representing x fields (independent variables), this will give each class C_y the probability:

$$p(C_y \mid f_1, \ldots, f_x)$$

To estimate the parameters of the NB model in our approach, we use the maximum likelihood method which can be written as follows:

$$p(C_y \mid f) = \frac{p(C_y)\, p(f \mid C_y)}{p(f)}$$

which can otherwise be understood as:

$$posterior\ probability = \frac{prior * likelihood}{evidence}$$

We apply the NB classifier over sequential titled time windows of the joint stream S, using interleaved test then train chunks [32], where each window-batch S_T of the arriving joint stream S is used first for testing, then it is used for training the classifier. Next, for each window-batch S_T, we calculate the joint probabilities. The joint probability is a measure that calculates the likelihood of two or more items occurring together at the same time point, and it can be expressed in our approach as:

$$p(C_y) \prod_{i=1}^{x} p(f_i \mid C_y)$$

Naive Bayes has been reported in the literature as one of the ideal algorithms for stream mining, due to its incremental nature; incremental learners cope with the concept drift faster than other learners [22] [33]. Furthermore, the NB is the only generative classification model, it provides a complete model of the probabilistic structure of the data; this will help to identify the interesting patterns in our approach [22].

Furthermore, to overcome the minority class problem which occurs when the minority class instance does not appear in the training set and appears in the test set, we seek to modify the training set of the classifier. In particular, as we mentioned above, every window-batch (S_T) is used first for testing then for training, so once the window-batch (S_T) is used for testing, we seek to identify the incorrectly classified instances, then we store the minimum number of these instances in a window (W_{min}), and include them in the training set of the subsequent windows, and we update the window (W_{min}) with the arrival of new window-batch (S_{T+1}). This will help to "be aware" of the exceptional situations in the parking lots and maintaining them for future reference.

C. Support Definition

Support definition is a significant part in our pattern discovery approach. We aim to develop a method to calculate a dynamic support threshold which detects the interesting patterns, including the rare patterns, from the imbalanced evolving streams. To be more precise, in many real-world application streams, some items appear very frequently in the data set, while others rarely appear, therefore mining frequent patterns using a fixed support threshold from these streams is insufficient; choosing a low threshold value will identify a large number of meaningless patterns, whereas a high threshold value may omit the rare patterns.

Based on the definition of support in Section II-D, and the definition of the joint probability (JP) in Section III-B which indicates the occurrence of items together, we define the support threshold value as follows:

$$Sup_T = min(JPs_T)$$

So we select the minimum joint probabilities (calculated from the NB classifiers in the previous section) for each window batch (S_T) to be the support threshold value of this window (Sup_T), the support will be updated for each window-batch. The proposed dynamic threshold calculation method can be used with real-time or near real-time data since the calculations depend on the current and previous time windows (the previous time window is used to train the NB classifier).

D. Mining Imbalanced Evolving Streams

The next step is mining the frequent patterns in the joint stream S. Our approach is based on the FP-growth algorithm, which is used widely for static data, and in order to make it applicable to the imbalanced evolving streams, we propose two main modifications:

1) Using a dynamic support threshold (Sup_T), which has been explained in the previous section III-C, instead of the fixed threshold value.
2) Using the NB technique to construct and update the FP-tree. Different than the original FP-growth algorithm, which scans the data set twice, first to get the frequencies then to find the frequent patterns, we use the joint probabilities, which are calculated by the NB classifier in Section III-B to indicate the frequencies and construct the frequent items list. Accordingly, the frequent item list is constructed based on the classifier fields ($f_1, \ldots, f_x, class$) in a descending order, then we subsequently build the FP-tree, the tree of frequent patterns, where each path represents a pattern. The frequent item list and FP-tree are constructed for the first window S_{T1} then they are updated with the arrival of each window-batch S_{Ti}, this reduces the cost of time and memory needed to scan the dataset.

Using our approach, we are able to know what are the frequent patterns over the period T. The description of our proposed FP-EStream algorithm is illustrated in Algorithm 1.

Algorithm 1: FP-EStream algorithm for pattern discovery from imbalanced evolving streams

```
FP-EStream
  Input  : IoT − Stream : Joint stream of
           heterogeneous IoT streams
  Output: FPs : Frequent patterns from IoT − Stream
for each incoming window S_T in IoT − Stream do
    if S_T is the first window in the stream then
        NB − model = build-the-classifier (S_T )
    else
        NB − model = update-the-classifier (S_T )
    end
    NB − model (S_T)
    Ps_T= prior probabilities from NB − model (S_T)
    CPs_T = the conditional probabilities from
      NB − model (S_T)
    JPs_T = calculate-the-joint-probabilities(P, CP)
    Sup_T = find-minimum-joint-probability(JPs_T)
    if FIL = ∅ then
        FIL = build-frequent-item-list( JPs)
        Build FP − tree
    else
        Update FIL
        Update FP − tree
    end
    FPs (S_T) = mining-the-frequent-patterns (Sup_T,
      FP − tree, S_T )
end
```

IV. Experimental Work

A. Data Set Description

In this paper, we apply our approach on the IoT streams in the parking lots setting at a typical University and we have used data from the University of Essex to construct the experimental setting, where there are six different parking lots, and only three main entries to all parking lots as follows:

- The first entry leads to the North car park, which is designated for both staff members and students, also it has special parking spaces for drop-off for the University's nursery, and a drop off/collection point for taxis.
- The second entry leads to the Multi-deck car park which is intended for staff members only, and car park B which is used by gym members in addition to staff and students. Also, there is a drop-off/collection point for taxi at the car park B.
- The third entry leads to car park A which is specified for students, the Valley car park which has special parking spaces for visitors and the Constable Building parking lot which is designated for on-campus hotel staff and the hotel customers/visitors.

A map of the University's parking lots is shown in Fig. 2.

The Multi-deck car park is the only parking lot which is equipped with a camera sensor for number plate recognition,

Fig. 2. Parking lots map of University of Essex

however, the IoT trend makes us believe that in the future a lot more places will be equipped with such sensors, and for the purpose of this work we assume that there is a camera sensor at each entry to sense the cars that cross the entry point.

In this scenario, the drivers, who are looking for parking spaces, are divided into four groups: staff members, students, gym members and visitors; staff members and students get a discounted rate on the parking fee. Moreover, there is a smart pay station located at each parking lot, where the drivers can pay the parking fee, or they can alternatively pay for parking using a (presumed) mobile phone application.

In fact, to get a real understanding of the situation in the parking lots, we used the following techniques:

- We have approached and had in-depth discussions regarding the operation of the car parks with the Transport Policy Manager of the University of Essex.
- We arranged an hour duration meetings with two of the traffic officers of the University of Essex to obtain a better understanding of the observed patterns from their experience as they patrol the parking lots on a daily basis and ensure that users have paid for parking.
- We have obtained and examined a report of the daily usage of the multi-deck car park for three months of the Summer term of 2016.

The parking charges are applied from 9:00 am until 16:00 from Monday to Friday, otherwise it is free to park, and if a driver (a staff member or student) registers his/her car with the Estates Management Help Desk, s/he is entitled to pay a cheaper parking ticket charge, 10 pence per hour or 70 pence per day, otherwise s/he pays the normal visitors' rate.

There are 1620 parking spaces, where the total number of registered users-of both staff members and students- for the year 2015-16 was 2942 which is double the number of spaces available in the parking lots, and the number of registered users as staff members is 1613, exceeding the number of registered students for 2015-16. Moreover, from the report analysis, we found that 8:00 and 9:00 in the morning are the peak hours, then the number of cars arriving decreases slightly; this is reasonable considering the typical start of the working day and therefore lectures, classes and other events. Also, there is a significant increase in the number of cars on Tuesdays and conversely a significant decrease on Fridays. Furthermore, according to the interviews with the traffic officers, there is a slight decrease in the number of cars at parking lots especially in car park A and Valley car park towards the end of term, while in the summer, car park A is nearly empty and the Valley car park does not reach its full capacity and there is a noticeable increase in the number of visitors. However, there is no clear difference between term time and summer in other parking lots where most of the users are staff members.

B. Model Simulation and Pre-processing

As we mentioned earlier, the car park lots in the University of Essex are not currently equipped with sensors throughout, so we have not had access to a full real-dataset from heterogeneous data streams for our work. Moreover, we searched for data sets with a similar streaming environment, but we could not find a sufficient one, in fact, such datasets are not publicly available due to privacy concerns.

In order to undertake our experiments, we first conducted an analysis of all the available data (discussion, interviews and reports) to understand seasonal patterns of behaviour, and subsequently we constructed a simulation model in order to generate artificial data and emulate the car parking environment (enriched with additional sensors). We chose the simulator NetLogo to simulate drivers' behaviour in the parking lots. NetLogo is a multi-agent system based model which uses its own programming language to create models [36]. We are not going to discuss the details of the simulation in this paper, due to space limitations, however, it is important to note that we use the multinomial distribution to allocate cars into parking lots, and since we do not have the precise parking probability in each parking lot, we calculate the initial parking probability based on the number of registered drivers. Screen shots of the simulation are shown in Fig 3.

There are different payment schemes to pay for parking and apart from paying on a daily basis, staff members may

Fig. 3. The graphic user interface of the simulation in NetLogo

opt to pay for the car park annually. Based on the information from the Estates Management Helpdesk, around 40% of drivers pay the annual or term fee and do not use the daily payment machine. Therefore, we have assumed these drivers are using the mobile application, so we can get their payment information through the mobile data stream, this will be used to complement the payment data stream from the pay stations.

To apply our proposed algorithm, first, we choose the Autumn term which consists of 11 weeks, and we consider a working day to be from 7:00 in the morning until 16:00 in the afternoon (cleaners and maintenance staff tend to start their shifts earlier than 9:00 am). As we are working based on a University environment where the lectures are blocks of one-hour, we select one-hour windows long to be the optimal time-window length to apply our algorithm. When nine one-hour windows are accumulated, a one day window will be formed, similarly, five days from Monday to Friday, will create one week window, and the 11-weeks window will form one-term window, so we will deal with a maximum of 51 windows per week. Then, we simulated the three streams: the sensor stream, and the payment data stream from both the pay stations and the mobile application. After that, we joined the streams over the term window based on the car plate number using MS-SQL, then we removed the rows that contain null values and derived the parking duration from the payment information. Moreover, as we have chosen the NB to be the classifier in our model, and in order to prepare the data stream for the next step, we converted the entry time into three intervals: peak, off-peak and high-peak. We also converted the parking duration into categorical features; it is common practice for NB techniques to discretise all numeric features so that all features are categorical [23].

C. Extracting Patterns from Imbalanced Evolving Streams

After joining the data streams, we seek to classify the arriving cars to the parking lots, we want to know the type of driver based on the collected parking information (parking location and parking duration). Accordingly, the base learner - the NB - is applied on the joint stream S. We implemented it using (naivebayes) package in R [35] and the MOA software framework [32] that contains implementations of several state-of-the-art classifiers and evaluation methods.

Furthermore, we use interleaved test then train one-hour windows, where each window (S_T) is used first for testing, then with the arrival of a new window, it is used for training. There are two common approaches to evaluate the classifiers in the streaming data: hold out (including the hold out set on periodical intervals), and the predictive sequential (Prequential) evaluation [14][34]. While each example or window of examples is first used to test the current model, then it is used for training in the Prequential evaluation, only some instances of the stream are used for testing in the periodical hold out. We tested both approaches, and as expected, the interleaved test then train chunks gives us better performance metrics than the periodical hold out. Table I shows the performance difference between the two evaluation methods. It is clear that not only

TABLE I
COMPARISON BETWEEN THE TWO EVALUATIONS METHODS

	Accuracy	**Kappa**
Interleaved chunks	99.82	99.63
Periodical hold out	79.59	51.62

the accuracy of the interleaved test then train chunks is higher than the periodical hold out, but also that the Kappa measure is much better. Based on the NB model, we calculate the joint probabilities, and consequently we select the minimum value of these probabilities to be the support threshold for extracting the frequent patterns for this window.

In order to discover the frequent patterns, we use the calculated dynamic threshold with FP-growth-itemsets-with-strings algorithm from SPMF library (the Java open-source pattern mining library) [37]. Accordingly, we construct the frequent items list for the first window from the joint probabilities, based on the classifier attributes (sensor, parking location, driver type) in descending order and build the FP-tree as well. After that, we used the (rJava) package to integrate Java code (modifed FP-growth algorithm) with the R implementation (classification model). The screen shot in Fig 4, shows the emerging patterns between 09:00 AM and 10:00 AM on Monday in the first week of October (the autumn term).

In addition, in Fig 5, we can see the difference in the emerging patterns between the beginning of the work-day, from 08:00 to 09:00 in the morning, and the end of the work-day, from 14:00 to 15:00 in the afternoon, for the same Monday. Also, we can see the change in the emerging patterns between the begining and the end of the term in Fig 6, which shows the emerging patterns on Thursday for the same interval, from 10:00 AM to 11:00 AM, of the first and the last week of the Autumn term. By investigating the obtained patterns over the different intervals: daily, weekly and over the Autumn term, it is apparent these patterns are consistent with those in the real data (from interviews and reports). For example, as Fig 5 shows, there is a significant decrease in the cars arrival rate in the afternoon. Also, as shown in Fig 6, there is a considerable drop in the cars arrival rate at the end of the term, particularly in car park A (the student car park).

D. Results and Discussion

In order to assess our method, we evaluate the classifier performance, then we evaluate the frequent pattern mining

```
========== FPM-Stream - STATS ==========
 Transactions count from this window : 503
 Frequent itemsets count : 11
 Total time ~ 1 ms
=================================================
Sensor  Park    Driver Frecuency
1  S2 BPark   Student   9
2  S1 VPark   Visitor  10
3  S3 NPark   Student  12
4  S1 CPark     Staff  14
5  S2 BPark GymMember  18
6  S1 APark   Student  28
7  S1 VPark     Staff  45
8  S2 BPark     Staff  62
9  S3 NPark     Staff  96
10 S1 VPark   Student  97
11 S2 MPark     Staff 112
```

Fig. 4. The emerging patterns over one-hour window

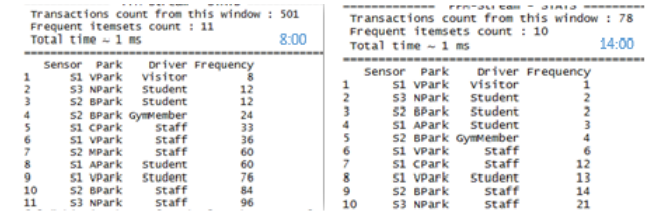

Fig. 5. The change in the emerging patterns over one day

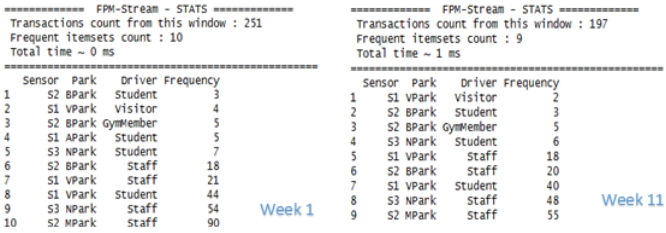

Fig. 6. The change in the emerging patterns over the term

performance and speed. Firstly, we want to show how the proposed method of applying adaptive learner and modifying the training set by keeping window of misclassified examples affect the learning from the imbalanced evolving streams. Accordingly, we compared the majority class algorithm using interleaved chunks with the NB algorithm, using the MOA framework over the whole dataset. From Fig 7, it can be seen that by far the highest performance is for the NB algorithm, this provides evidence for efficiency of applying the adaptive learner to cope with the concept drift in the multi-class imbalance streaming scenarios.

Then we compared the two algorithms' outputs, the NB algorithm with the modified training set and the NB algorithm without modifying the training set, with the ground truth, by comparing the predicted target values of each classifier with the true target values. We chose three days interval from Monday to Wednesday of the first week of the Autumn term 2015-2016. It is apparent from the data in Table II and Table III that very few instances which the NB only algorithm classified them inaccurately, this number was reduced using the NB algorithm with the modified training set. In more detail, on Tuesday between 11:00 and 12:00, when some students park

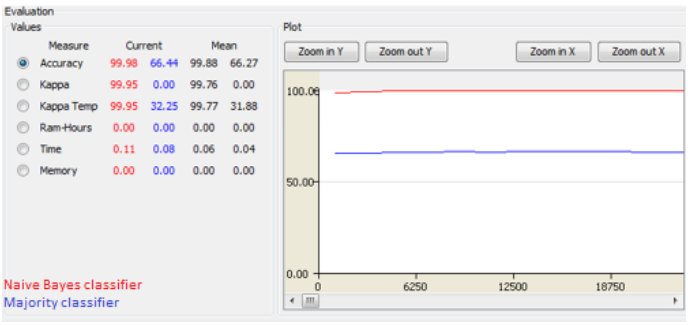

Fig. 7. The comparison between the two classification techniques

in the Multi-deck car park, both classifiers failed to classify six instances of class (student) and classified them as (staff), the reason is the Multi-deck car park is designated for staff-member use, so students rarely park there (only when other parking lots are full). However, when this occurs again on Wednesday between 10:00 and 11:00 (after nine window-batches), the NB algorithm with the modified training set was able to detect the students who park in the multi-deck car park, and it classified them correctly, where the NB only algorithm classified them inaccurately again. Typically, techniques that learn from the evolving stream are focusing on the most recent data (previous window) as it is time consuming to train the learner on all data items which have already been seen. As far as we know, this is the first time that has been shown experimentally that we can simply learn from the unbalanced evolving streams and detect the minority classes' instances in these streams by applying adaptive learner approaches and by modifying the training set.

Secondly, to validate the effectiveness of the dynamically calculated support threshold in our method, we examined the detected patterns from the joint stream S of IoT streams using a low fixed threshold value, a high fixed threshold value, and our dynamically calculated threshold over different window-batches. We will demonstrate some windows in the investigated dataset, compare the detected patterns using each threshold and discuss the findings. Accordingly, we chose nine consecutive windows from 07:00 until 16:00 on Wednesday of the first week of the term in the joint stream S, and we tried to detect the frequent patterns over these window-batches using the following techniques:

1) FP-growth algorithm and here we selected a low fixed threshold ($Sup_T = zero$)
2) FP-growth algorithm and here we selected a high fixed threshold ($Sup_T = 0.05$)
3) Our proposed technique FP-EStream where the support threshold is dynamically calculated ($Sup_T \in [0.001, \ldots, 0.01]$)

TABLE II
CONFUSION MATRIX FOR THE NB CLASSIFIER ONLY

Predicted class	Actual class			
	Staff	Student	Gym member	Visitor
Staff	6872	11	0	1
Student	0	2936	0	0
Gym member	0	0	401	0
Visitor	0	0	0	141

TABLE III
CONFUSION MATRIX FOR THE NB WITH THE MODIFIED TRAINING SET

Predicted class	Actual class			
	Staff	Student	Gym member	Visitor
Staff	6872	6	0	1
Student	0	2941	0	0
Gym member	0	0	401	0
Visitor	0	0	0	141

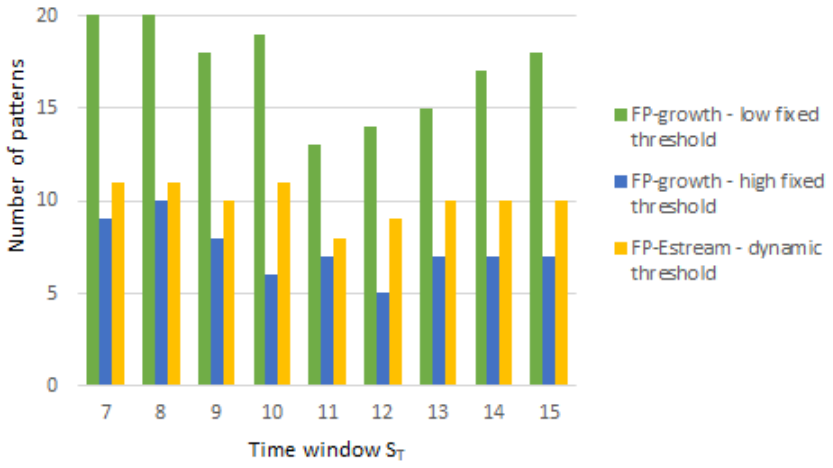

Fig. 8. The comparison between FP-EStream and FP-growth

We use the SPMF library to apply the FP-growth algorithm on the joint stream S, and to get comparable results, the input for the FP-growth is the same input which we used for the classifier in the proposed FP-EStream technique. A comparison between the three methods is shown in Fig. 8. We can clearly see that the number of patterns extracted by the low threshold (0) are nearly double the numbers of patterns extracted using the dynamic threshold, however, the extracted patterns using threshold (0) are insignificant patterns and require an additional pruning step. By contrast, the numbers of patterns extracted by the high threshold (0.05) are less than the number of patterns extracted using the dynamic one, the high threshold failed to detect any pattern relating to the visitors, the gym-members or car park A. In general, choosing a fixed low threshold increased the number of detected patterns, these expanded patterns require an additional filtering step. On the other hand, choosing a high threshold not only decreased the number of detected patterns, but also failed to detect all rare patterns. It is clear from the graph, that the number of patterns discovered by our proposed dynamic threshold is in between the two previous algorithms, it avoids too many insignificant patterns and also enables the detection of rare patterns. Interestingly this threshold identified all the patterns which are detected by the lower threshold zero including the rare patterns (no filter needed).

The speed is the main performance measure for mining the frequent patterns. Different from the multiple conditional tree generate-and-test approach, the classifier in our method helps to narrow the number of derived patterns down to only the interesting patterns, without repeated scans. Hence, reducing the cost of the database scan and pruning search will speed up the mining process. The average time required to discover patterns from one-hour window of the first week of the Autumn term using the FP-EStram is 5 ms, faster than the average time required over the same interval using the FP-growth 11 ms (threshold = 0). Also we needed 7.162 sec to apply our algorithm over the whole term window, compared to 14.268 sec to apply the FP-growth. This may scale up to show the efficiency of applying the FP-EStram algorithm on larger datasets and real time processing.

V. Conclusions

Discovering patterns from dynamic streams which have skewed distributions is challenging and learning techniques are required to process data in real-time, using minimal computational resources, adapt with the concept drift and cope with imbalanced class distributions. Despite the problem of class imbalance having been studied in the context of existing streams mining research, only few works have explored the multi-class imbalance problem in evolving streams.

In this work, we develop a method called FP-EStream to capture the dynamic of patterns from heterogeneous streams. The contribution of this work is three-fold. Firstly, we apply an adaptive base classifier in the multi-class imbalance case to cope with the concept drift. We use the NB to classify the multiple streams over a sequence of titled time windows. Secondly, we keep a window of instances which are inaccurately classified, and use it to modify the training set over the subsequent windows, to discover the minority classes' instances. Thirdly, we propose a dynamically calculated support threshold to identify the patterns from the heterogeneous streams and detect the rare patterns. The results have shown the usability of our method to identify the frequent patterns from the IoT streams. This highlights the efficiency of applying adaptive learner approaches using modified training set to cope with the concept drift in the multi-class imbalance scenarios. In addition, the results support the merit of using our dynamic support threshold to overcome the rare patterns problem in non-stationary streams with multi-class imbalance. While ensemble techniques are the most common approach to cope with multi-class imbalance problem in evolving streams, this work is the first

work, to our knowledge, which has shown experimentally the efficiency of extending other concept drift adaption techniques to the multi-class imbalanced evolving streams context. Also, our method does not need to generate candidates or scan the database many times, unlike other pattern discovery approaches (e.g., FP-growth). This framework can be adapted and used in different domains of application for discovering patterns in IoT streams (e.g., tube fault diagnosis). For future works, we aim to show the applicability of our method in different domains such as water monitoring, where we have meters, sensors, and open-data sources. Our method may be applied on the collection of these streams to detect anomalous behaviour patterns (e.g. leakages), react quickly to critical situations and support predictive maintenance.

Acknowledgement The authors would like to acknowledge the support of the Business and Local Government Data Research Centre (grant number ES/L011859/1) funded by the Economic and Social Research Council (ESRC) for undertaking this work.

REFERENCES

[1] V. Lopez, S. del Ro, J. M. Bentez, and F. Herrera, Cost-sensitive linguistic fuzzy rule based classification systems under the mapreduce framework for imbalanced big data, Fuzzy Sets and Systems, pp. 5-38, 2015.

[2] Louis Columbus, "Internet of things forecasts.", 2017 [Online]. Available: https://www.forbes.com/sites/louiscolumbus/2017/12/10/2017-roundup-of-internet-of-things-forecasts/#68374bc11480/.[Accessed 5-March-2018].

[3] A. Bifet, G. de Francisci Morales, J. Read, G. Holmes, and B. Pfahringer, Efficient online evaluation of big data stream classifiers, in Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp. 59-68, ACM, 2015.

[4] J. Han, J. Pei, and Y. Yin, Mining frequent patterns without candidate generation, in ACM sigmod record, vol. 29, pp. 1-12, ACM, 2000.

[5] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Wozniak, Ensemble learning for data stream analysis: A survey, Information Fusion, vol. 37, pp. 132-156, 2017.

[6] D. Brzezinski and J. Stefanowski, Prequential auc for classifier evaluation and drift detection in evolving data streams, in International Workshop on New Frontiers in Mining Complex Patterns, pp. 87-101, Springer, 2014.

[7] S. Wang, L. L. Minku, and X. Yao, Dealing with multiple classes in online class imbalance learning., in IJCAI, pp. 2118-2124, 2016.

[8] D. Brzezinski, J. Stefanowski, A. Nienkotter, X. Jiang, M. Last, M. Stoliar, M. Friedman, R. Cornelisse, S. Choenni, M. Munir, et al., Ensemble classifiers for imbalanced and evolving data streams,

[9] D. Brzezinski and J. Stefanowski, Prequential auc: properties of the area under the roc curve for data streams with concept drift, Knowledge and Information Systems, vol. 52, no. 2, pp. 531-562, 2017.

[10] R. Agrawal, R. Srikant, et al., algorithms for mining association rules," in Proc. 20th int. conf. very large data bases, VLDB, vol. 1215, pp. 487-499, 1994.

[11] B. Liu, W. Hsu, and Y. Ma, Mining association rules with multiple minimum supports, in Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 337- 341, ACM, 1999.

[12] S.-M. Cheng and M.-Y. Day, Technologies and Applications of Artificial Intelligence: 19th International Conference, TAAI 2014, Taipei, Taiwan, November 21-23, 2014, Proceedings, vol. 8916. Springer, 2014.

[13] J. Gama and M. M. Gaber, Learning from data streams: processing techniques in sensor networks. Springer, 2007.

[14] J. Gama, R. Sebastiao, and P. P. Rodrigues, On evaluating stream learning algorithms, Machine learning, vol. 90, no. 3, pp. 317-346, 2013.

[15] Y. Sun, Z. Wang, H. Liu, C. Du, and J. Yuan, Online ensemble using adaptive windowing for data streams with concept drift, International Journal of Distributed Sensor Networks, vol. 12, no. 5, p. 4218973, 2016.

[16] S. Mohamad, Active learning for data streams. PhD thesis, Bournemouth University, 2017.

[17] T. S. Sethi and M. Kantardzic, On the reliable detection of concept drift from streaming unlabeled data, Expert Systems with Applications, vol. 82, pp. 77-99, 2017.

[18] B. Krawczyk and M. Wozniak, One-class classifiers with incremental learning and forgetting for data streams with concept drift, Soft Computing, vol. 19, no. 12, pp. 3387-3400, 2015.

[19] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, A survey on concept drift adaptation, ACM computing surveys (CSUR), vol. 46, no. 4, p. 44, 2014.

[20] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, Learning with drift detection, in Brazilian symposium on artificial intelligence, pp. 286- 295, Springer, 2004.

[21] A. Bifet and R. Gavalda, Learning from time-changing data with adaptive windowing, in Proceedings of the 2007 SIAM international conference on data mining, pp. 443-448, SIAM, 2007.

[22] J. N. van Rijn, G. Holmes, B. Pfahringer, and J. Vanschoren, The online performance estimation framework: heterogeneous ensemble learning for data streams, Machine Learning, pp. 1-28, 2018.

[23] R. N. Lichtenwalter and N. V. Chawla, Learning to classify data streams with imbalanced class distributions, New Frontiers in Applied Data Mining. LNCS. Springer, Heidelberg, 2009.

[24] Wang, Shuo, Leandro L. Minku, and Xin Yao, "A learning framework for online class imbalance learning," Computational Intelligence and Ensemble Learning (CIEL), 2013 IEEE Symposium on , pp. 36-45, 2013.

[25] B. Mirza, Z. Lin, J. Cao, and X. Lai, Voting based weighted online sequential extreme learning machine for imbalance multi-class classification, in Circuits and Systems (ISCAS), 2015 IEEE International Symposium on, pp. 565-568, IEEE, 2015.

[26] J. Hipp, U. Guntzer, and G. Nakhaeizadeh, Algorithms for association rule mininga general survey and comparison, ACM sigkdd explorations newsletter, vol. 2, no. 1, pp. 58-64, 2000.

[27] C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu, Mining frequent patterns in data streams at multiple time granularities, Next generation data mining, vol. 212, pp. 191-212, 2003.

[28] R. U. Kiran and M. Kitsuregawa, Mining correlated patterns with multiple minimum all-confidence thresholds, in Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 295-306, Springer, 2013.

[29] K. P. Lakshmi and C. Reddy, Efficient classifier generation over stream sliding window using associative classification approach, International Journal of Computer Applications, vol. 115, no. 22, 2015.

[30] C. K.-S. Leung and B. Hao, Mining of frequent itemsets from streams of uncertain data, in Data Engineering, 2009. ICDE09. IEEE 25th International Conference on, pp. 1663-1670, IEEE, 2009.

[31] S. Vanamala, L. P. Sree, and S. D. Bhavani, Rare association rule mining for data stream, in Computer and Communications Technologies (ICCCT), 2014 International Conference on, pp. 1-6, IEEE, 2014.

[32] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, Moa: Massive online analysis, Journal of Machine Learning Research, vol. 11, no. May, pp. 1601-1604, 2010.

[33] J. Read, A. Bifet, B. Pfahringer, and G. Holmes, Batch-incremental versus instance-incremental learning in dynamic and evolving data, in International Symposium on Intelligent Data Analysis, pp. 313-323, Springer, 2012.

[34] J. Gama, R. Sebastiao, and P. P. Rodrigues, Issues in evaluation of stream learning algorithms, in Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 329-338, ACM, 2009.

[35] R. C. Team et al., R: A language and environment for statistical computing, 2013.

[36] S. Tisue and U. Wilensky, Netlogo: A simple environment for modeling complexity, in International conference on complex systems, vol. 21, pp. 16-21, Boston, MA, 2004.

[37] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.-W. Wu, and V. S. Tseng, Spmf: a java open-source pattern mining library, The Journal of Machine Learning Research, vol. 15, no. 1, pp. 3389-3393, 2014.