Credit Card Fraud Detection and Concept-Drift Adaptation with Delayed Supervised Information

Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi and Gianluca Bontempi

Abstract—Most fraud-detection systems (FDSs) monitor streams of credit card transactions by means of classifiers returning alerts for the riskiest payments. Fraud detection is notably a challenging problem because of concept drift (i.e. customers’ habits evolve) and class unbalance (i.e. genuine transactions far outnumber frauds). Also, FDSs differ from conventional classification because, in a first phase, only a small set of supervised samples is provided by human investigators who have time to assess only a reduced number of alerts. Labels of the vast majority of transactions are made available only several days later, when customers have possibly reported unauthorized transactions. The delay in obtaining accurate labels and the interaction between alerts and supervised information have to be carefully taken into consideration when learning in a concept-drifting environment.

In this paper we address a realistic fraud-detection setting and we show that investigator’s feedbacks and delayed labels have to be handled separately. We design two FDSs on the basis of an ensemble and a sliding-window approach and we show that the winning strategy consists in training two separate classifiers (on feedbacks and delayed labels, respectively), and then aggregating the outcomes. Experiments on large dataset of real-world transactions show that the alert precision, which is the primary concern of investigators, can be substantially improved by the proposed approach.

Index Terms—Fraud Detection, Concept Drift, Unbalanced Data, Data Streams, Anomaly Detection.

I. INTRODUCTION

Everyday a huge and growing number of credit cards payments takes place while being targeted by fraudulent activities. Companies processing electronic transactions have to promptly detect any fraudulent behavior in order to preserve customers’ trust and the safety of their own business.

Most fraud-detection system (FDSs) employ machine-learning algorithms to learn frauds’ patterns and detect them as datastreams of transactions come [4]. In particular, we focus here on FDSs which aim to detect frauds by means of classifiers that label transactions as fraudulent or genuine. Fraud detection is particularly challenging for two reasons [5]: frauds represent a small fraction of all the daily transactions [3] and their distribution evolves over time because of seasonality and new attack strategies [29]. This situation is typically referred to as concept drift [19] and is of extreme relevance for FDSs which have to be constantly updated either by exploiting the most recent supervised samples or by forgetting outdated information that might be no more useful whereas not misleading.

In a real-world setting, it is impossible to check all transactions. The cost of human labour seriously constrains the number of alerts, returned by the FDS, that can be validated by investigators. Investigators in fact check the alerts by calling the cardholders, and then provide the FDS with feedbacks indicating whether the alerts were related to fraudulent or genuine transactions. These feedbacks, which refer to a tiny fraction of the daily transactions amount, are the only real-time information that can be provided to train or update classifiers. The labels of the rest of transactions can be assumed to be known several days later, once a certain reaction-time for the customers have passed: all the transactions that customers do not report as frauds are considered genuine. In the paper we will distinguish between immediate feedback samples (i.e. transactions annotated with the investigator feedback) and delayed samples, whose labels is obtained only after some time. This distinction is crucial for the design of an accurate FDS, though most FDSs in the literature [25], [36], [16], [4] assume an immediate and accurate labeling after the processing of each transaction. This oversimplifying assumption ignores the alert-feedback interaction, which makes the few recent supervised couples dependent from the performance of the FDS itself.

Another substantial difference between the real-world settings and the ideal ones considered in literature is that the primary concern of any FDS should be to return a small number of very precise alerts, then reducing the number of genuine transactions (false positives) that have be controlled by investigators. In practice, the optimal FDS should be the one maximizing the number of frauds detected within the budget of alerts that can be reported. Notwithstanding, classical performance metrics considered in the literature are the area under the curve (AUC), the cost (namely, financial losses arising from misclassification), and metrics based on the confusion matrix [23] (e.g the F-measure), which are not necessarily meaningful for the alert precision.

In this work we show that, in a real-world fraud-detection scenario, it is convenient to handle immediate feedbacks separately from delayed supervised samples. The former, in fact, are selected as the most risky transactions according to the FDS itself, while the latter refer to all the occurred transactions. Our claim is better illustrated in Section IV, where
we investigate two traditional learning approaches for FDSs, namely, i) a sliding-window approach where a classifier is retrained everyday on the most recent supervised samples and ii) an ensemble approach where, everyday, a new component replaces the oldest one in the ensemble. We designed and assessed two different solutions for each approach: in the first, feedbacks and delayed supervised samples are pooled together while in the second we train two distinct classifiers, based on feedbacks and delayed samples respectively, and then aggregate the outputs. Experiments shown in Section V on two real-world credit card datasets indicate that handling feedbacks separately from delayed training samples can substantially improve the alert precision. We motivate this result as the fact that this solution guarantees a prompter reaction to concept drift: additional experiments on datasets that have been manipulated to introduce concept drift in specific days, confirm our intuition.

To the best of our knowledge, this is also the first work addressing the problem of fraud detection when supervised pairs are provided according to the alert-feedback interaction, as formulated in Section III.

II. RELATED WORKS

FDSs are confronted with two major challenges: i) handling non-stationary streams of transactions, namely a stream where the statistical properties of both frauds and genuine transactions change overtime; ii) handling the class unbalance, since legitimate transactions generally far outnumber the fraudulent ones. In what follows we provide an overview of state-of-the-art FDSs with a specific focus on solutions for evolving and unbalanced data streams.

In the fraud-detection literature both supervised [7], [10], [4] and unsupervised [6], [34] solutions have been proposed. Unsupervised methods do not rely on transactions labels (i.e. genuine or fraudulent) and associate fraudulent behaviours [6] to transactions that do not conform with the majority. Unsupervised methods exploit clustering algorithms [31], [36] to group customers into different profiles and identify frauds as transactions departing from customer profile (see also the recent survey by Phua [30]).

In this paper we will focus on supervised methods. Supervised methods exploit labels that investigators assign to transactions for training a classifier and, during operation, detect frauds by classifying each transaction in the incoming stream [5]. Fraud detection has been often considered as an application scenario for several classification algorithms, e.g. Neural networks [22], [1], [16], [7], Support Vector Machines [37], Decision Trees [13] and Random Forest [12]).

Learning on the stream of credit transactions is a challenging issue because transactions evolve and change over time, e.g. customers’ behaviour change in holiday seasons and new fraud activities may appear. This problem is known as concept drift [19] and learning algorithms operating in non-stationary environments typically rely only on the supervised information that is up-to-date (thus relevant), and remove any obsolete training sample [2]. Most often, concept-drift adaptation is achieved by training a classifier over a sliding window of the recent supervised samples (e.g. STAGGER [32] and FLORA [38]) or by ensemble of classifiers where recent supervised data are used to train a new classifier while obsolete ones are discarded (e.g. SEA [33] and DWM [26]).

Streams of credit card transactions present an additional challenge: the classes are extremely unbalanced since frauds are typically less than 1% of genuine transactions [13]. Class unbalance is typically addressed by resampling methods [24], which balance the training set by removing samples of the majority class (undersampling) or by replicating the minority class (oversampling). In practice, concept-drift adaptation in an unbalanced environment is often achieved by combining ensemble methods and resampling techniques. The class unbalance problem is addressed in [20], [21] by propagating minority class training samples and undersampling the majority class. Chen and He proposed REA [11] where they recommend to propagate only examples from the minority class that belong to the same concept using a k-nearest neighbors algorithm. Learn++NIE [15] creates multiple balanced training sets from a batch using undersampling, then it learns a classifier on each balanced subset and combines all classifier’s predictions. Lichtenwalter and Chawla [28] suggest to propagate not only positives, but also observations from the negative class that are misclassified in the previous batch to increase the boundary definition between the two classes.

All the aforementioned learning frameworks demand a training set of recent instances with their own ground-truth class label. However, in a real-world FDS, this is often not possible because only few recent supervised couples are provided according to the alert-feedback interaction described in Section I. The only FDS explicitly handling concept drift in the transaction streams is [35] which nevertheless, like other FDS presented in the literature [6], [7], [10], ignores the alert-feedback interaction.

It is worth to remark that this alert-feedback interaction could remind an active-learning scenario where the learner is allowed to query an oracle for requiring informative supervised couples from a large set of unlabelled observations. Unfortunately in a FDS scenario, this solution is not feasible since an exploration phase, where investigators should check a large number of (possibly interesting) transactions, would not be considered as acceptable.

III. PROBLEM FORMULATION

We formulate here the fraud detection problem as a binary classification task where each transaction is associated to a feature vector \( x \) and a label \( y \). Features in \( x \) could be the transaction amount, the shop id, the card id, the timestamp or the country, as well as features extracted from the customer profile. Because of the time-varying nature of the transactions’ stream, typically, FDSs train (or update) a classifier \( K_t \) every day \( t \). The classifier \( K_t : \mathbb{R}^n \to \{+, -\} \) associates to each feature vector \( x \in \mathbb{R}^n \), a label \( K_t(x) \in \{+, -\} \), where \(+\) denotes a fraud and \( -\) a genuine transaction. Since frauds represent a negligible fractions of
the total number of transactions, the positive class is also called the minority class and the negative one the majority class.

In general, FDSs operate on a continuous stream of transactions because frauds have to be detected online, however, the classifier is updated once a day, to gather a sufficient amount of supervised transactions. Transactions arriving at day \( t \), namely \( T_t \), are processed by the classifier \( \mathcal{K}_{t-1} \) trained in the previous day (\( t-1 \)). The \( k \) riskiest transactions of \( T_t \) are reported to the investigators, where \( k > 0 \) represents the number of alerts the investigators are able to validate. The reported alerts \( A_t \) are determined by ranking the transactions of \( T_t \) according to the posterior probability \( P_{\mathcal{K}_{t-1}}(+|x) \), which is the estimate, returned by \( \mathcal{K}_{t-1} \), of the probability for \( x \) to be a fraud. The set of reported alerts at day \( t \) is defined as

\[
A_t = \{ x \ s.t. \ r(x) \leq k \}
\]

where \( r(x) \in \{1,\ldots,\#T_t\} \) is the rank of the transaction \( x \) according to \( P_{\mathcal{K}_{t-1}}(+|x) \), and \( \#(\cdot) \) denotes the cardinality of a set. In other terms, the transaction with the highest probability ranks first (\( r(x) = 1 \)) and the one with the lowest probability ranks last (\( r(x) = \#T_t \)).

Investigators will then provide feedbacks \( F_t \) about the alerts in \( A_t \), defining a set of \( k \) supervised couples \((x, y)\)

\[
F_t = \{(x, y), \ x \in A_t \},
\]

which represents the only immediate information that the FDS receives. At day \( t \), we also receive the labels of all the transactions processed at day \( t-\delta \), providing a set of delayed supervised couples \( D_{t-\delta} = \{(x, y), \ x \in T_{t-\delta} \} \), see Figure 1. Though these transactions have not been personally checked by investigators, they are by default assumed to be genuine after \( \delta \) days, as far as customers do not report frauds.\(^1\) As a result, the labels of all the transactions older than \( \delta \) days are provided at day \( t \). The problem of receiving delayed labels is also referred to as verification latency \([27]\).

It is worth to remark that this is still a simplified description of the processes regulating companies analyzing credit card transactions. For instance, it is typically not possible to extract the alerts \( A_t \) by ranking the whole set \( T_t \), since transactions have to be immediately passed to investigators; similarly, delayed supervised couples \( D_{t-\delta} \) do not come all at once, but are provided over time. Notwithstanding, we deem that the most important aspects of the problem (i.e. the alert-feedback interaction and the time-varying nature of the stream) are already contained in our formulation and that further details would unnecessarily make the problem setting complex.

Feedbacks \( F_t \) can either refer to frauds (correct alerts) or genuine transactions (false alerts): correct alerts are the true positives (TP), while false alerts are the false positives (FP). Similarly, \( D_{t-\delta} \) contains both fraud (false negative) and genuine transactions (true negatives), although the vast majority of transactions belong to the genuine class. Figure 2 illustrates the two types of supervised pairs that are provided everyday.

The goal of a FDS is to return accurate alerts: when too many FPs are reported, investigators might decide to ignore forthcoming alerts. Thus, what actually matters is to achieve the highest precision in \( A_t \). This precision can be measured by the quantity

\[
p_k(t) = \frac{\#\{(x, y) \in F_t \ s.t. \ y = +\}}{k}
\]

where \( p_k(t) \) is the proportion of frauds in the top \( k \) transactions with the highest likelihood of being frauds \([4]\).

**IV. Learning Strategy**

The fraud-detection scenario described in Section III suggests that learning from feedbacks \( F_t \) is a different problem than learning from delayed samples in \( D_{t-\delta} \). The first difference is evident: \( F_t \) provides recent, up-to-date, information while \( D_{t-\delta} \) might be already obsolete once it comes. The second difference concerns the percentage of frauds in \( F_t \) and \( D_{t-\delta} \). While it is clear that the class distribution in \( D_{t-\delta} \) is always skewed towards the genuine class (see Figure 2), the number of frauds in \( F_t \) actually depends on the performance of classifier \( \mathcal{K}_{t-1} \): values of \( p_k(t) \sim 50\% \) provide feedbacks \( F_t \) where frauds and genuine transactions are balanced, while
high precision values might even result in $F_t$ skewed towards frauds. The third, and probably the most subtle, difference is that supervised couples in $F_t$ are not independently drawn, but are instead selected by $K_{t-1}$ among those transactions that are more likely to be frauds. As such, a classifier trained on $F_t$ learns how to label transactions that are most likely to be fraudulent, and might be in principle not precise on the vast majority of genuine transactions. Therefore, beside the fact that $F_t$ and $D_{t-\delta}$ might require different resampling methods, $F_t$ and $D_{t-\delta}$ are also representative of two different classification problems and, as such, they have to be separately handled. In the following, two traditional fraud-detection approaches are presented (Section IV-A), and further developed to handle separately feedbacks and delayed supervised couples (Section IV-B). Experiments in Section V show that this is a valuable strategy, which substantially improves the alert precision.

A. Conventional Classification Approaches in FDS

During operation, feedbacks $F_t$ and delayed supervised samples $D_{t-\delta}$ can be exploited for training or updating the classifier $K_t$. In particular, we train the FDS considering the feedbacks from the last $\delta$ days (i.e. $\{F_t, F_{t-1}, \ldots, F_{t-(\delta-1)}\}$) and the delayed supervised pairs from the last $\alpha$ days before the feedbacks, i.e. $\{D_{t-\delta}, D_{t-\delta+1}, \ldots, D_{t-(\delta+\alpha-1)}\}$ (see Figure 2).

In the following we present two conventional solutions for concept-drift adaptation [34], [20] built upon a classification algorithm proving an estimate of the probability $P(\cdot\mid x)$.

- $\mathcal{W}_t$: a sliding window classifier that is daily updated over the supervised samples received in the last $\delta + \alpha$ days, i.e. $\{F_t, F_{t-1}, F_{t-2}, \ldots, D_{t-\delta}, \ldots, D_{t-(\delta+\alpha-1)}\}$ (see Figure 3).
- $\mathcal{E}_t$: an ensemble of classifiers $\{M_1, M_2, \ldots, M_\alpha, F\}$, where $M_i$ is trained on $D_{t-\delta-i}$ and $F$ is trained on all the feedbacks of the last $\delta$ days $\{F_t, F_{t-1}, \ldots, F_{t-(\delta-1)}\}$. The estimate of posterior probability $P_{\mathcal{E}_t}(\cdot\mid x)$ is estimated by averaging the posterior probabilities of the individual classifiers, $P_{\mathcal{E}_t}(\cdot\mid x)$.

These solutions implement two basic approaches for handling concept drift that can be further improved by adopting dynamic sliding windows or adaptive ensemble sizes [17].

B. Separating delayed Supervised Samples from Feedbacks

Our intuition is that feedbacks and delayed transactions have to be treated separately because, beside requiring different tools for handling class unbalance, they refer to different classification problems. Therefore, at day $t$ we train a specific classifier $\mathcal{F}_t$ on the feedbacks of the last $\delta$ days $\{F_t, F_{t-1}, F_{t-2}, \ldots, F_{t-(\delta-1)}\}$ and denote by $P_{\mathcal{F}_t}(\cdot\mid x)$ its posterior probability. We then train a second classifier on the delayed samples by means either of a sliding-window or an ensemble mechanism (see Figure 3). Let us denote by $\mathcal{W}_t^D$ the classifier trained on a sliding window of delayed samples $\{D_{t-\delta}, \ldots, D_{t-(\delta+\alpha-1)}\}$ and by $P_{\mathcal{W}_t^D}(\cdot\mid x)$ its posterior probabilities, while $\mathcal{E}_t^D$ denotes the ensemble of $\alpha$ classifiers $\{M_1, M_2, \ldots, M_\alpha\}$ where each individual classifier $M_i$ is trained on $D_{t-\delta-i}$, $i = 1, \ldots, \alpha$. Then, the posterior probability $P_{\mathcal{E}_t^D}(\cdot\mid x)$ is obtained by averaging the posterior probabilities of the individual classifiers.

Each of these two classifiers has to be aggregated with $\mathcal{F}_t$ to exploit information provided by feedbacks. However, to raise alerts, we are not interested in aggregation methods at the label level but rather at the posterior probability level. For the sake of simplicity we adopt the most straightforward combination approach based on averaging the posterior probabilities of the two classifiers ($\mathcal{F}_t$ and one among $\mathcal{W}_t^D$ and $\mathcal{E}_t^D$). Let us denote by $A_t^W$ the aggregation of $\mathcal{F}_t$ and $\mathcal{E}_t^D$ where $P_{A_t^W}(\cdot\mid x)$ is defined as:

$$P_{A_t^W}(\cdot\mid x) = \frac{P_{\mathcal{F}_t}(\cdot\mid x) + P_{\mathcal{E}_t^D}(\cdot\mid x)}{2}$$

Similar definition holds for the aggregation of $\mathcal{F}_t$ and $\mathcal{W}_t^D$ ($A_t^W$). Note that $\mathcal{F}_t$ and $\mathcal{W}_t^D$ jointly use the training set of $\mathcal{W}_t$ and, similarly, the two classifiers $\mathcal{F}_t$ and $\mathcal{E}_t^D$ jointly use the same training samples of $\mathcal{E}_t$ (see Figure 3).

However, in $\mathcal{W}_t$ feedbacks represent a small portion of the supervised samples used for training, hence they have little influence on $P_{\mathcal{W}_t}(\cdot\mid x)$, while in the aggregation $A_t^W$ their contribution becomes more prominent. Similarly, $\mathcal{F}_t$ represents one of the classifiers of the ensemble $\mathcal{E}_t$; hence it has in principle the same influence as all the other $\alpha$ classifiers.

There is no point of storing feedbacks from $F_{t-\delta}$ (or before), as these supervised couples are provided in $D_{t-\delta}$ (or before).
trained on delayed samples to determine $P_E(\cdot | x)$.\(^3\)

Experiments in Section V show that handling feedbacks separately from delayed supervised samples provides much more precise alerts, and that FDSs relying on classifiers trained exclusively on feedbacks and delayed supervised samples (like $A_F^W$ and $A^E_F$) substantially outperform FDSs trained on feedbacks and delayed supervised samples pooled together (like $W_t$ and $E_t$). In what follows, as practical example of the separation of feedbacks from delayed supervised couples, we detail the specific solutions based on Random Forests that were used in our experiments.

C. Two Specific FDSs based on Random Forest

As a base algorithm the FDSs presented in the previous section we used a Random Forest [9] with 100 trees. In particular, for $W_t^D$, $W_t$ and for all $M_i, i = 1, \ldots, \alpha$, we used a Balanced Random Forest (BRF) where each tree is trained on a balanced bootstrap sample, obtained by randomly undersampling the majority class while preserving all the minority class samples in the corresponding training set. Each tree of BRF receives a different random sample of the genuine transactions and the same samples from the fraud class in the training set, yielding a balanced training set. This undersampling strategy allows one to learn trees with balanced distribution and to exploit many subsets of the majority class. At the same time, this resampling method reduces training sizes and improve detection speed. A drawback of undersampling is that we are potentially removing relevant training samples form the dataset, however this problem is mitigated by the fact that we learn 100 different trees. Using undersampling allows us to rebalance the batches without propagating minority class observations along the streams as in [20]. Propagating frauds between batches should be avoided whenever possible, since it requires access to previous batches that we might not be able to store when data arrives in streams. In contrast, for $F_t$ that is trained on feedbacks we adopted a standard Random Forest (RF) where no resampling is performed.

V. EXPERIMENTS

We considered two datasets of credit card transactions of European cardholders: the first one (referred to as 2013) is composed of daily transactions from the 5\(^{th}\) of September 2013 to the 18\(^{th}\) of January 2014, the second one (referred to as 2014) contains transactions from the 5\(^{th}\) of August to the 9\(^{th}\) of September 2014. In the 2013 dataset there is an average of 160k transactions per day and about 304 frauds per day, while in the 2014 dataset there is on average 173k transactions and 380 frauds everyday. Table I reports few additional details about these datasets and shows that they are also heavily unbalanced.

In the first experiments we process both datasets to assess the importance of separating feedbacks from delayed supervised samples. Though we expect these streams to be

\(^3\)In the specific case of the ensemble and of posterior probabilities computed by averaging (4), the aggregation of $F_t$ and $E^D_t$ ($A^E_F$) corresponds to assigning a larger weight to $F_t$ in $E_t$.

\(^4\)We ran several experiments with $\alpha = 1, 8, 16, 24$ and found $\alpha = 16$ as a good trade-off between performance, computational load, and the number of days that can be used for testing in each stream.

<table>
<thead>
<tr>
<th>Table I</th>
<th>Datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>Start day</td>
</tr>
<tr>
<td>2013</td>
<td>2013-01-01</td>
</tr>
<tr>
<td>2014</td>
<td>2014-08-05</td>
</tr>
</tbody>
</table>
of times that one strategy is superior to the others.

A. Experiments on 2013 and 2014 Datasets

In order to evaluate the benefit of learning on feedbacks and delayed samples separately, we first compare the performance of classifier $\mathcal{W}_t$ against $F_t$, $\mathcal{W}_t^D$ and the aggregation $A_t^W$. Table II shows the average $p_k$ over all the batches for the two datasets separately. In both 2013 and 2014 datasets, $A_t^W$ outperforms the other FDSs in terms of $p_k$. The barplots of Figure 5 show the sum of ranks for each classifier and the results of the paired t-tests. Figure 5 indicates that in both datasets (Figures 5(a) and 5(b)) $A_t^W$ is significantly better than all the other classifiers. $F_t$ achieves higher average $p_k$ and higher sum of ranks than $\mathcal{W}_t^D$ and $\mathcal{W}_t$: this confirms that feedbacks are very important to increase $p_k$. Figure 4(a) displays the value of $p_k$ for $A_t^W$ and $\mathcal{W}_t$ in each day, averaged in a neighborhood of 15 days. During December there is a substantial performance drop, that can be seen as a Concept Drift (CD) due to a change in cardholder behaviour before Christmas. However, $A_t^W$ dominates $\mathcal{W}_t$ along the whole 2013 dataset, which confirms that a classifier $A_t^W$ that learns on feedbacks and delayed transactions separately outperforms a classifier $\mathcal{W}_t$ trained on all the supervised information pooled together (feedbacks and delayed transactions).

Figures 5(c), 5(d) and Tables III confirm this claim also when the FDSs implements an ensemble of classifiers. In particular, Figure 4(b) displays the smoothed average $p_k$ of classifiers $E_t^W$ and $E_t$. For the whole dataset $E_t^W$ has better $p_k$ than $E_t$.

B. Experiments on artificial dataset with CD

In this section we artificially introduce an abrupt CD in specific days by juxtaposing transactions acquired in different times of the year. Table IV reports the three datasets that have been generated by concatenating batches of the dataset 2013 with batches from 2014. The number of days after concept drift is set such that the FDS has the time to forget the information from the previous concept.

Table IV shows the values of $p_k$ averaged over all the batches in the month before the change for the sliding window approach, while Table V shows $p_k$ in the month after the CD. $A_t^W$ reports the highest $p_k$ before and after CD. Similar results are obtained with the ensemble approach.

Table V(a) shows the values of $p_k$ averaged over all the batches in the month before and after the change for the sliding window approach. For the whole dataset $A_t^W$ is superior than $E_t$.

![Fig. 4. Average $p_k$ per day (the higher the better) for classifiers on dataset 2013 smoothed using moving average of 15 days. In the sliding window approach classifier $A_t^W$ has higher $p_k$ than $\mathcal{W}_t$, and in the ensemble approach $A_t^F$ is superior than $E_t$.](image)

![Diagram](image)

### TABLE II

<table>
<thead>
<tr>
<th>Dataset 2013</th>
<th>Dataset 2014</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F$</td>
<td>mean</td>
</tr>
<tr>
<td>$\mathcal{W}_t^D$</td>
<td>0.609</td>
</tr>
<tr>
<td>$\mathcal{W}_t$</td>
<td>0.540</td>
</tr>
<tr>
<td>$A_t^W$</td>
<td>0.697</td>
</tr>
</tbody>
</table>

### TABLE III

<table>
<thead>
<tr>
<th>Dataset 2013</th>
<th>Dataset 2014</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{W}_t^D$</td>
<td>mean</td>
</tr>
<tr>
<td>$\mathcal{W}_t$</td>
<td>0.540</td>
</tr>
<tr>
<td>$E_t$</td>
<td>0.555</td>
</tr>
<tr>
<td>$A_t^F$</td>
<td>0.683</td>
</tr>
</tbody>
</table>

### TABLE IV

<table>
<thead>
<tr>
<th>Id</th>
<th>Start 2013</th>
<th>End 2013</th>
<th>Start 2014</th>
<th>End 2014</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD1</td>
<td>2013-09-05</td>
<td>2013-09-30</td>
<td>2014-08-05</td>
<td>2014-08-31</td>
</tr>
</tbody>
</table>

### TABLE V

<table>
<thead>
<tr>
<th>(a) Before CD</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F$</td>
</tr>
<tr>
<td>$\mathcal{W}_t^D$</td>
</tr>
<tr>
<td>$\mathcal{W}_t$</td>
</tr>
<tr>
<td>$A_t^W$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(b) After CD</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F$</td>
</tr>
<tr>
<td>$\mathcal{W}_t^D$</td>
</tr>
<tr>
<td>$\mathcal{W}_t$</td>
</tr>
<tr>
<td>$A_t^W$</td>
</tr>
</tbody>
</table>

Please note that classifier $F_t$ returns different results between Table II and Table III because of the stochastic nature of RF.

5Please note that classifier $F_t$ returns different results between Table II and Table III because of the stochastic nature of RF.
that $F_t$ often outperforms $W_t^D$ (and $\mathcal{E}_t^D$), and $\mathcal{W}_t$ (and $\mathcal{E}_t$). We deem that $F_t$ outperforms $W_t^D$ (resp. $\mathcal{E}_t^D$) since $W_t^D$ (resp. $\mathcal{E}_t^D$) are trained on less recent supervised couples. As far as the improvement with respect to $\mathcal{W}_t$ (and $\mathcal{E}_t$) is concerned, our interpretation is that this is due to the fact that $\mathcal{W}_t$ (and $\mathcal{E}_t$) are trained on the entire supervised dataset, then weakening the specific contribution of feedbacks.

Our results instead show that aggregation prevents the large amount of delayed supervised samples to dominate the small set of immediate feedbacks. This boils down to assign larger weights to the most recent than to the old samples, which is a golden rule when learning in non-stationary environments. The aggregation $A_t^W$ is indeed an effective way to attribute higher importance to the information included in the feedbacks. At the same time $A_t^E$ is a way to balance the contribution of $F_t$ and the remaining $\alpha$ models of $E_t$.

Another motivation of the accuracy improvement is that classifiers trained on feedbacks and delayed samples address two different classification tasks (Section IV). For this reason too, it is not convenient to pool the two types of supervised samples together.

Finally, the aggregation presented in equation 4 provides equal weights to the two posterior probabilities $P_{F_t}$ and $P_{W_t^D}$ ($P_{E_t^D}$). However, more sophisticated and eventually adaptive aggregation schemes (e.g. non-linear or stacking [8]) could be used to react to concept drift. In fact, in a rapidly drifting environment, the relative weight of $P_{F_t}$ should eventually increase, because $W_t^D$ (or $E_t^D$) might be obsolete and prone to false alarms.

VI. Conclusion

In this paper we formalise a framework that reproduces the working conditions of real-world FDSs. In a real-world fraud-detection scenario, the only recent supervised-information is provided on the basis of the alerts generated by the FDS and feedbacks provided by investigators. All the other supervised samples are provided with a much larger delay.

Our intuition is that the alert-feedback interaction has to be explicitly considered to improve alert precision and
that feedbacks and delayed samples have to be separately handled when training a realistic FDS. To this purpose, we have considered two general approaches for fraud detection: a sliding window and an ensemble of classifiers. We have then compared FDSs that separately learn on feedbacks and delayed samples against FDSs that pool all the available supervised information together. Experiments run on real-world streams of transactions show that the former strategy provides much more precise alerts than the latter, and that it also adapts more promptly in concept-drifting environments.

Future work will focus on investigating adaptive mechanisms to aggregate the classifier trained on feedbacks and the one trained on delayed samples, to further improve the alert precision in non-stationary streams.

REFERENCES


