Expert Systems With Applications 100 (2018) 234–245

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Sequence classification for credit-card fraud detection

Johannes Jurgovsky[a,*], Michael Granitzer[a], Konstantin Ziegler[a], Sylvie Calabretto[b], Pierre-Edouard Portier[b], Liyun He-Guelton[c], Olivier Caelen[d]

[a] University of Passau, Faculty of Computer Science and Mathematics, Germany
[b] INSA Lyon, Laboratoire d'Informatique en Image et Systèmes d'Information, France
[c] ATOS Worldline, R&D High Processing & Volume Division, France
[d] ATOS Worldline, R&D High Processing & Volume Division, Belgium

ARTICLE INFO

Article history:
Received 12 September 2017
Revised 2 January 2018
Accepted 23 January 2018
Available online 31 January 2018

Keywords:
Finance
Credit-card fraud detection
Sequence classification
Long short-term memory networks

ABSTRACT

Due to the growing volume of electronic payments, the monetary strain of credit-card fraud is turning into a substantial challenge for financial institutions and service providers, thus forcing them to continuously improve their fraud detection systems. However, modern data-driven and learning-based methods, despite their popularity in other domains, only slowly find their way into business applications.

In this paper, we phrase the fraud detection problem as a sequence classification task and employ Long Short-Term Memory (LSTM) networks to incorporate transaction sequences. We also integrate state-of-the-art feature aggregation strategies and report our results by means of traditional retrieval metrics.

A comparison to a baseline random forest (RF) classifier showed that the LSTM improves detection accuracy on offline transactions where the card-holder is physically present at a merchant. Both the sequential and non-sequential learning approaches benefit strongly from manual feature aggregation strategies. A subsequent analysis of true positives revealed that both approaches tend to detect different frauds, which suggests a combination of the two. We conclude our study with a discussion on both practical and scientific challenges that remain unsolved.

1. Introduction

As businesses continue to evolve and migrate to the internet and money is transacted electronically in an ~~ever growing cashless banking economy~~, accurate fraud detection remains a key concern for modern banking systems. It is not only to limit the direct losses incurred by fraudulent transactions, but also to ensure that legitimate customers are not adversely impacted by automated and manual reviews. In payment industry, fraud on card occurs when someone steals information from your card to do purchases without your permission and the detection of these fraudulent transactions has become a crucial activity for payment processors.

A typical fraud detection systems is composed of an *automatic tool* and a *manual process*. The *automatic tool* is based on fraud detection rules. It analyzes all the new incoming transactions and assigns a fraudulent score. The *manual process* is made by fraud investigators. They focus on transactions with high fraudulent score and provide a binary feedback (fraud or genuine) on all the transactions they analyzed.

The fraud detection systems can be based on *expert* driven rules, *data* driven rules or a combination of both types of rules. *Expert* driven rules try to identify specific scenarios of fraud discovered by the fraud investigators. A scenario of fraud can be "*a cardholder does a transaction in a given country and, in the two next weeks, (s)he does another transaction for a given amount in another given country.*" If this scenario is detected in the stream of transactions, then the fraud detection system will produce an alert. *Data* driven rules are based on machine learning algorithms. They learn the fraudulent patterns and try to detect them in a data-stream of new incoming transactions. The very commonly employed machine learning algorithms include logistic regression, support vector machines and random forests (Bahnsen, Aouada, Stojanovic, & Ottersten, 2016; Bhattacharyya, Jha, Tharakunnel, & Westland, 2011). Fraud detection is a challenging machine learning problem

Abbreviations: LSTM, Long short-term memory network; RF, Random forest; ECOM, e-commerce transactions or online transactions; F2F, Face-to-face transactions or offline transactions.

[*] Corresponding author.

E-mail addresses: johannes.jurgovsky@uni-passau.de (J. Jurgovsky), michael.granitzer@uni-passau.de (M. Granitzer), mail@zieglerk.net (K. Ziegler), sylvie.calabretto@insa-lyon.fr (S. Calabretto), pierre-edouard.portier@insa-lyon.fr (P.-E. Portier), liyun.he-guelton@worldline.com (L. He-Guelton), olivier.caelen@worldline.com (O. Caelen).

https://doi.org/10.1016/j.eswa.2018.01.037
0957-4174/© 2018 Elsevier Ltd. All rights reserved.

for many reasons (Dal Pozzolo, Boracchi, Caelen, Alippi, & Bontempi, 2018): (i) the data distribution evolves over time because of seasonality and new attack strategies, (ii) fraudulent transactions represent only a very small fraction of all the daily transactions and (iii) the fraud detection problem is intrinsically a sequential classification task. In this paper, we address primarily the last issue by using LSTM networks as machine learning algorithms in fraud detection systems.

We bring together recent contributions made in the domains of machine learning and credit-card fraud detection and show through a comprehensive study how a data-driven fraud detection system can benefit from these advancements. We provide empirical results on a real-world credit-card transaction dataset and study the classification accuracy on online (e-commerce) and offline (point-of-sale or face-to-face) transactions independently. Also, we discuss typical pitfalls encountered in the application of a sequence learner, such as an LSTM, in the context of credit-card fraud detection. In particular, we make the following contributions:

- We compare a sequence learner (Long Short-Term Memory) with a static learner (random forest) on a real-world fraud detection dataset.
- We study the added benefit of state-of-the-art manual feature engineering on both learning approaches.
- We show that information about the transaction history, induced either through a sequence learner or feature engineering, strongly improves the fraud detection accuracy on offline transactions. However on online transactions, information induced through a sequence learner adds no benefit over a static learner.
- We show that frauds detected by an LSTM are consistently different from the frauds detected by a random forest. This holds on both online and offline transactions.

2. Credit-card fraud detection

Besides the interest of financial institutions in mitigating their financial losses, credit-card fraud detection has become an attractive test-bed for data mining researchers to study a broad range of interacting properties that rarely arise altogether in a single application domain. These properties include imbalanced classes, temporal dependence between samples, concept drift, dynamic misclassification cost, real-time detection, etc. Likewise, a plethora of methods from supervised learning, unsupervised learning, anomaly detection and ensemble learning lend themselves to address individual or several of these properties (Abdallah, Maarof, & Zainal, 2016; Carneiro, Figueira, & Costa, 2017; Phua, Lee, Smith, & Gayler, 2010).

Supervised learning methods treat fraud detection as a binary classification problem and they aim to model the conditional distribution over the two classes *fraud* and *genuine* given training data. In a systematic review of 49 journal articles (Ngai, Hu, Wong, Chen, & Sun, 2011) showed that Decision Trees, Neural Networks, Logistic Regression and Support Vector Machines have been the methods of choice among the many available methods. Bhattacharyya et al. (2011) compared the predictive accuracy of Logistic Regression, random forests and Support Vector Machines on a real-world credit-card fraud data set under varying proportions of fraud in the training set, finding that random forests demonstrated an overall higher precision at low recall levels.

Most studies address the imbalanced data problem by incorporating a sampling technique prior to the training process on the data level. Minority class oversampling, majority class undersampling or a mixture of both called SMOTE (Chawla, Bowyer, Hall, & Kegelmeyer, 2002) have proven to be effective ways to balance the data for data mining algorithms. Alternatively, the imbalance problem can be addressed on an algorithmic level. Cieslak, Hoens, Chawla, and Kegelmeyer (2012) proposed to use the Hellinger distance as splitting criterion in decision trees. Since the class priors do not appear explicitly in this distance, the class imbalance ratio does not influence the calculation. The authors demonstrate empirically the superiority of this method on imbalanced data and competitive performance on balanced datasets, thus, effectively bypassing the necessity of sampling. Dal Pozzolo and Bontempi (2015) applied this technique successfully to credit-card fraud detection. Besides the class imbalance problem, fraud detection algorithms also need to account for the fact that the class conditional distribution changes over time. Dal Pozzolo et al. (2018) revealed the presence of such *concept drift* by comparing the predictive accuracy of a static classifier that is trained once and never updated against an ensemble of classifiers that is iteratively re-trained according to several proposed updating schedules.

Another direction of research focuses on integrating the dynamic misclassification costs into the learning algorithm. Unlike other cost-sensitive learning problems, the cost matrix in fraud detection depends not only on the true class and the predicted class but also on the particular transaction under consideration. A common strategy is to set the misclassification cost proportional to the available credit limit on the card to put more weight on the first frauds in a sequence. Sahin, Bulkan, and Duman (2013) and Bahnsen, Aouada, and Ottersten (2015) inject the cost-sensitivity in decision trees via a modified splitting criterion that minimizes the sum of misclassification costs. Mahmoudi and Duman (2015) use Fisher Discriminant Analysis for credit-card fraud detection and they account for the dynamic misclassification cost by adding an example-dependent weight in the computation of the classes' sample means.

Anomaly detection methods take a different perspective. Such methods typically learn a model of normal instances, i.e. genuine transactions, and then identify those transactions that do not conform to the model as anomalies. These methods require the definition of a suitable anomaly score that quantifies the degree of an instance to be abnormal. Anomaly scores can be derived directly from the likelihood in probabilistic models, from the average path lengths in ensembles of decision trees (Liu, Ting, & Zhou, 2008), the local density in the neighbourhood around an instance (Breunig, Kriegel, Ng, & Sander, 2000; Zhao, Zhang, & Qin, 2017) or the support of infrequent item sets obtained via Association Rule Mining (Hemalatha, Vaidehi, & Lakshmi, 2015).

In credit-card fraud detection, a fraud is not exclusively a property of the transaction itself but rather a property of both the transaction and the particular context it appeared in, i.e. the account and the merchant. In recent years, there has been a surge of interest in characterizing this context in form of a purchase history or some kind of activity record that enables a classifier to better identify fraudulent transactions within the recent genuine activity. Therefore, in the following two sections, we review two strategies from literature that allow us to construct such context descriptions.

2.1. Feature engineering for temporal sequences

When constructing a credit-card fraud detection system, the choice of features is crucial for accurate classification. Unsurprisingly, considerable research effort has been dedicated to the development of expressive features. As Bahnsen et al. (2016) note, across many credit-card datasets the set of attributes is very similar, comprising attributes such as time, amount, merchant category, account number, transaction type, etc. In a ”traditional” fraud detection system these attributes are used directly as input features to train a classifier on the binary classification problem. Such systems operate on the transaction level by considering transactions in isolation and ignore the fact that the frequency or volume of transactions at different time periods and merchants bears rich information about each individual account.

One graph-theoretic approach for extracting expressive features is discussed in Van Vlasselaer et al. (2015). Aside from the detection system they maintain a network graph in which nodes correspond to merchants or card holders and edges correspond to transactions between these entities. The edge weight corresponds to the volume of transactions transmitted between a pair of entities and it decays with time. From the graph they extract network features that measure the exposure of each entity to fraud. These features include a score for the card holder, the merchant and the transaction, aggregated over short, medium and long time periods.

The more traditional approaches aim to extract features that indicate the degree to which the current transaction differs from previous transactions. Whitrow, Hand, Juszczak, Weston, and Adams (2009) proposed feature aggregation strategies to summarize the historic purchase activities of card holders. The authors create an activity record for each transaction and account by summing up the amounts spent in previous transactions within time periods spanning the past 1, 3 and 7 days. The record contains several of such aggregate variables – one for each value of a categorical feature. For instance, the aggregate variable MCC1434_3 would denote the sum of amounts of all transactions from the previous 3 days, in which the merchant category code was 1434. For all transactions they compute the corresponding activity record and feed the new feature vector to a classifier, that is trained to predict whether the status of the account is *compromised* or *normal*. Their experiments show that, across many different classifiers, the prediction accuracy significantly increased as the aggregation spanned more than a single transaction – with random forests clearly performing best. Since then, the proposed aggregation strategy served as a generic template for developing new features (Jha, Guillen, & Westland, 2012; Krivko, 2010).

In Bahnsen et al. (2016), the authors use such variables aggregated over the past 24 h as features in a classification setting to study the performance of different classifiers and the impact of the features on monetary savings.

2.2. Sequence classification

Sequential learning is the study of learning algorithms for sequential data. The context information and the sequential dependency between data points are taken into account at the algorithmic level. A general review on the properties of sequential classification tasks and suitable methods to cope with them can be found in Dietterich (2002). These methods include sliding window methods, recurrent sliding windows or conditional random fields.

While sliding window based approaches tend to ignore the sequential relationship between data points inside the windows, a better solution is to resort to model-based approaches that assume explicitly a sequential dependency between consecutive data points. In its simplest form such model could be a Markov chain defined on the data points. However, in many applications of practical interest the sequential dependency is presumably more evident or useful as a sequence of latent, so called *hidden*, states that control the sequence of observed data points. Since these states act like some form of memory of the past, a decision about a sequence of data points can be reduced to a decision based on the states.

There are different types of hidden state architecture. In the family of probabilistic models, Hidden Markov models (HMM) are one of the most popular representatives of generative models. They can be used to perform unsupervised learning of normal or abnormal succession patterns in sequential data. In credit-card fraud detection a typical use-case is to run a new transaction sequence as query against the model and compute the likelihood of whether the sequence could have been generated from the normal or abnormal HMM. The likelihood is then used as a score to effectively perform anomaly detection (Srivastava, Kundu, Sural, & Majumdar, 2008). An alternative hidden state architecture is the Conditional Random Field (CRF) (Lafferty, McCallum, & Pereira, 2001). Such a model is trained discriminatively with labels in order to directly predict the conditional distribution over classes given a sequence of data points.

Although conceptually similar, a Recurrent Neural Network (RNN) is a hidden state architecture from the family of non-probabilistic models. The connectivity between hidden states, so called nodes in neural network terminology, also satisfies the Markov assumption. RNNs are trained discriminatively to predict the label of a transaction given the sequence of transactions from the past. Graves (2012) give a detailed introduction to RNNs and their variants. More recently, one of the variants, Long Short-Term Memory Networks (LSTM), received much attention due to its capability to learn long term dependencies in sequences. This property led to broad success in practical applications such as speech recognition and machine translation (Graves & Jaitly, 2014; Sutskever, Vinyals, & Le, 2014).

Aside from Wiese and Omlin (2009), sequence classification seems to be a yet unexplored terrain in the credit card fraud detection community.

3. Methodology

Depending on the perspective we take, fraudulent transactions can be understood as anomalies in the purchase behavior of customers or as a set of transactions that form a class in their own right which is opposite to the class of genuine transactions.

Either way, in feature space frauds very well intermingle with genuine transactions because of two reasons. First, genuine purchase actions from millions of customers naturally cover a broad spectrum of variability. And secondly, fraudsters apply various hard to detect, yet rational, strategies to execute fraudulent acts that span several customer accounts across varying time periods – but in the end, these acts will likewise only appear as individual transactions in a dataset. Thus, identical purchase actions can at the same time reflect either completely legitimate behavior in the context of some customers or obvious anomalies in the context of others.

In order to better support discrimination between transactions that are hard to distinguish, we identified two approaches that enable us to summarize the transaction history of customers and to use this summary during the classification of individual transactions. With the first method, we focus on recovering the sequential structure of a customer's transaction history by modeling the transition dynamics between transactions with a Recurrent Neural Network (Section 3.1). The second method is a well-established practice in the domain of credit-card fraud detection and it is based on manual feature engineering (Section 3.2).

3.1. Recurrent neural network for sequence classification

A Long Short-Term Memory Network (LSTM) is a special type of recurrent neural network (RNN). Recurrent neural networks have been developed in the 80s (Elman, 1990; Werbos, 1988; Williams & Hinton, 1986) to model time series data. The structure of a RNN is similar to that of a standard multilayer perceptron, with the addition that it allows connections among hidden units associated with discrete time steps. The time steps index the individual elements in a sequence of inputs. Through the connections across time steps the model can retain information about the past inputs, enabling it to discover temporal correlations between events that are possibly far away from each other in the input sequence. This is a crucial property for proper learning of time series where the occurrence

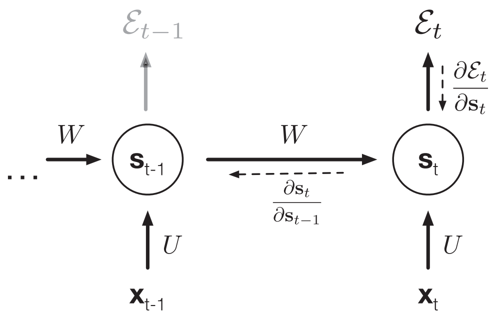


Fig. 1. Schematic of a recurrent neural network unrolled in time by creating a copy of the model for each time step.

of one event might depend on the presence of several other events further back in time.

A vanilla recurrent neural network is initialized with a state vector $\mathbf{s}_0$, usually consisting of only zeros, and then receives a sequence $\mathbf{x}_{1:\,T}$ of input vectors $\mathbf{x}_t$, indexed by positive integers t. In our experiments we fix the length of input sequences to either $T = 5$ (SHORT) or $T = 10$ (LONG). The RNN then runs through a sequence of state vectors $\mathbf{s}_t$, determined by the following recurrence (1)

$$\mathbf{s}_t = \mathbf{W}\sigma(\mathbf{s}_{t-1}) + \mathbf{U}\mathbf{x}_t + \mathbf{b}, \tag{1}$$

where the trainable parameters of the model are the recurrent weight matrix **W**, the input weight matrix **U** and the biases **b**. The hyper-parameters of the model are the dimensions of the vectors and matrices, and the non-linear element-wise activation function σ – hyperbolic tanh in our case. The mapping from an input sequence to a state sequence is typically referred to as a *recurrent layer* in neural network terminology. RNNs can have several of such archetypes stacked on top of each other (Section 4.3). For the training, we now additionally specify a cost function and a learning algorithm.

A cost function E_T measures the performance of the network on some given task after having seen T input vectors and transitioning to state $\mathbf{s}_T$. The distribution over classes *fraud* and *non-fraud* given state $\mathbf{s}_T$ is modeled with a logistic regression output model outputting predictions $\hat{y}_T = p(y_T|\mathbf{s}_T)$. We interpret the true label $y_t \in \{0, 1\}$ of a transaction as the probability of $\mathbf{x}_t$ to belong to class 0 or 1 and measure the cost induced by the model's predicted probabilities $\hat{y}_t$ by means of the cross-entropy error, defined as $E_T = L(\mathbf{x}_{1:T}, y_T) = -y_T \log \hat{y}_T - (1 - y_T)\log(1 - \hat{y}_T)$.

The model parameters $\theta = (\mathbf{W}, \mathbf{U}, \mathbf{b})$ are learned by minimizing the cost E_T with a gradient-based optimization method. One approach that can be used to compute the required gradients is Backpropagation Through Time (BPTT). BPTT works by unrolling a recurrent network in time to represent it as a deep multi-layer network with as many hidden layers as there are time steps (see Fig. 1). Then, the well-known Backpropagation algorithm (Williams & Hinton, 1986) is applied on the unrolled network.

While in principle, the recurrent network is a simple and powerful model, in practice, it is hard to train properly with gradient descent. Among the many reasons why this model is so unwieldy, there are two major problems that have been coined the *vanishing* and *exploding gradient* problem (Bengio, Simard, & Frasconi, 1994). These problems refer to the fact that gradients can either decrease or increase exponentially as the length of the input sequence grows. Very small gradients cause the network to learn very slowly or even stop completely. Very large gradients make the parameters diverge which leads to numerical problems and consequently derails learning.

With the recurrent connection between latent states, the parameters θ affect the error not only through the last but all previous states. Likewise, the error depends on **W** through all states **s**. This dependency becomes problematic when we compute the gradient of E_T w.r.t. θ:

$$\frac{\partial E_T}{\partial \theta} = \sum_{1 \le k \le T} \left(\frac{\partial E_T}{\partial \mathbf{s}_T} \frac{\partial \mathbf{s}_T}{\partial \mathbf{s}_k} \frac{\partial \mathbf{s}_k}{\partial \theta} \right) \tag{2}$$

The Jacobian matrix $\frac{\partial \mathbf{s}_T}{\partial \mathbf{s}_k}$ contains all component-wise interactions between state $\mathbf{s}_k$ and state $\mathbf{s}_T$. It can be understood as a means to transport back the error from state T to state k. It is given as a product of all pairwise interactions between consecutive states:

$$\frac{\partial \mathbf{s}_T}{\partial \mathbf{s}_k} = \prod_{k < i \le T} \frac{\partial \mathbf{s}_i}{\partial \mathbf{s}_{i-1}} \tag{3}$$

This product is the reason behind the difficulties to learn long-term dependencies with gradient-based optimization methods. Each factor $\frac{\partial \mathbf{s}_i}{\partial \mathbf{s}_{i-1}}$ involves both the recurrent weight matrix and the derivative of σ. The number of factors in (3) is $T - k$ and Pascanu, Mikolov, and Bengio (2013) showed that it is sufficient for the largest eigenvalue of the recurrent weight matrix to be smaller than 1 for long term components to decrease exponentially and necessary for it to be larger than 1 for gradients to increase exponentially with $T - k$.

Several solutions exist to address these problems. Using an L1- or L2-penalty on the recurrent weight matrix can ensure that the eigenvalue of largest magnitude never exceeds 1, given an initialization with sufficiently small weights. Another proposal is based on the assumption that if the model exhibits from the beginning the same kind of asymptotic behavior as the one required by the target, then it's less likely for gradients to explode (Doya, 1993). However, it is not trivial to initialize a model in this specific regime. Gradient-clipping is another rather practical approach that involves clipping element-wise components of the gradient when they exceed a fixed threshold (Mikolov, Deoras, Kombrink, Burget, & Černockỳ, 2011). Finally, a solution for avoiding the vanishing gradient problem was proposed by Hochreiter and Schmidhuber (1997). They removed the direct dependency of $\frac{\partial \mathbf{s}_i}{\partial \mathbf{s}_{i-1}}$ on the recurrent weight matrix (Bayer, 2015) by introducing memory cells. This modified network structure is called a Long Short-Term Memory Network and it constitutes the state of the art on many real world tasks such as speech recognition, hand writing recognition and statistical machine translation. A detailed exposition of the LSTM-network is beyond the scope of this work. Therefore, we forward the reader to the original article of Hochreiter and Schmidhuber (1997).

3.2. Feature engineering

As an alternative to modeling transaction sequences with an LSTM, we employ traditional feature engineering and, in particular, feature aggregations in order to explicitly summarize the purchase activities of card holders.

3.2.1. Time delta

A sequence learner detects patterns in sequences of consecutive transactions. We suppose these patterns resemble some form of latent purchase behaviour of card-holders. If so, the behavioral patterns should be invariant to the concrete points in time when the purchase actions were actually taken. To support a time normalization on input sequences that span very different time periods, we extract the time between two consecutive transactions in minutes and add it explicitly as additional feature:

$$\mathrm{tdelta}_i = x_i^{(Time)} - x_{i-1}^{(Time)} \tag{4}$$

3.2.2. Feature aggregations

An elegant way to extract information from the history of a card holder is to aggregate the values of certain variables along the transaction sequence. To construct such feature aggregations, we follow the procedure that was recently proposed by Bahnsen et al. (2016). This simple and yet powerful procedure can be considered the state of the art of feature engineering in credit-card fraud detection. To each transaction, we add new features according to some pre-defined rules. The value of a new feature is computed with an aggregation function applied to a subset of the preceding transactions. The goal is to create an activity record from the transaction history of a card-holder which quantifies the degree to which the current transaction differs from previous ones.

Let $(x_t)_{t\geq 1}$ be the temporally ordered sequence of transactions of a single card-holder. We denote the value of a particular variable in a transaction by superscript, e.g. $x_1^{(\text{Amt})}$ is the amount spent in the first transaction x_1. Based on a single transaction x_k, we select a subset of transactions from the past t_h hours according to sets of categorical variables $\{\}$, $\{A\}$ or $\{A, B\}$:

$$\begin{aligned}\mathcal{S}_k^{\{\}} &= \{x_i \,|\, (hours(x_i^{(Time)}, x_k^{(Time)}) < t_h)\}_{i<k} \\ \mathcal{S}_k^{\{A\}} &= \{x_i \,|\, (hours(x_i^{(Time)}, x_k^{(Time)}) < t_h) \wedge (x_i^{(A)} = x_k^{(A)})\}_{i<k} \\ \mathcal{S}_k^{\{A,B\}} &= \{x_i \,|\, (hours(x_i^{(Time)}, x_k^{(Time)}) < t_h) \wedge (x_i^{(A)} = x_k^{(A)}) \\ &\quad \wedge (x_i^{(B)} = x_k^{(B)})\}_{i<k} \qquad (5)\end{aligned}$$

Each set $\mathcal{S}_k$ comprises all transactions from the previous t_h hours before x_k, where the categorical variable took the same value as in x_k. The categorical variables and the time horizon t_h can be seen as constraints imposed on the subset. For instance, if we define $A := Country,\ B := Merchant$ and $t_h = 24$, the subset $\mathcal{S}_k^{\{A,B\}}$ contains all transactions from the past 24 h which were issued in the same country and at the same merchant as transaction x_k. Likewise, we can employ other categorical variables by adding further identity constraints to the boolean conjunction.

Now we can define aggregation functions on $\mathcal{S}_k$. There are many possibilities to define such functions and even though all of them might be equally valid, we restrict ourselves to the two functions that were proposed in the cited work: The total amount spent and the number of transactions.

$$\text{sum}_{\mathcal{S}_k} = \sum_{x \in \mathcal{S}_k} x^{(Amt)} \qquad (6)$$

$$\text{count}_{\mathcal{S}_k} = |\mathcal{S}_k| \qquad (7)$$

The pair of features $(\text{sum}_{\mathcal{S}_k}, \text{count}_{\mathcal{S}_k})$ corresponds to a particular choice of categorical variables and t_h. In order to cover a broader range of statistics from the transaction history, we compute such pairs for each element in the power set of {*country, merchant category, card entry mode*} and a time horizon of 24 h. Finally, we append all these pairs to the feature vector of transaction x_k.

3.3. E-commerce and face-to-face

As in any statistical modeling task, we can observe the true phenomena in the real world only through a proxy given as a finite set of point-wise observations. In credit-card fraud detection, the true phenomenon of interest is the genuine purchase behaviour of card holders or, likewise, the malicious behavior of fraudsters. Our assumption is that this object, which we loosely term the *behaviour*, is controlled by some latent, yet consistent, qualities. With its state variables, the LSTM is in principle capable of identifying these qualities from the sequence of observations.

In the real world, societal conventions, official regulations or plain physics impose constraints on the possible variability of observations and thereby on the complexity of the qualities that control them. For instance, opening hours strictly limit when and where customers are able to purchase their goods or services. Geographical distances and traveling modalities restrict the possibilities of consecutive transactions. We can expect that all the face-to-face transactions we see in our dataset respect, to some degree, these real world constraints. In contrast, e-commerce transactions or rather their corresponding online purchases are widely unconstrained, according to both time and place. There is hardly any attribute that could not genuinely change in arbitrary ways from one transaction to the next.

We hypothesize that the presence of real-world constraints in face-to-face transactions gives rise to more obvious behavioral patterns with less variation. In that case, a sequence learner should benefit from the more consistent sequential structure.

4. Experiments

Motivated by preceding statistical analyses and considerations regarding real-world purchase behavior (see Section 3.3), we separately study the impact of a sequence learner on the detection accuracy on e-commerce and on face-to-face transactions, respectively. We contrast the results with a non-sequence learner, a random forest classifier. In Section 4.1, we describe the dataset and in Section 4.2, we discuss the features of the records. In Section 4.3, we specify the employed classifiers and in Section 4.4 the evaluation metrics.

4.1. Datasets

Our study is based on a dataset of credit-card transactions, recorded from March to May 2015. Each transaction in the dataset has a boolean label assigned, indicating whether the transaction was in fact a fraudulent act. This labeling is performed by a team of human investigators who monitor in near-realtime the stream of transactions. They maintain and manage a pool of expert rules which raise alerts in the presence of anomalous behavior in predefined scenarios. Once an alert is raised, human investigators can contact the card-holder to check the validity of suspicious transactions. Accordingly, if card-holders discover mistakes in their credit card statement, the team of investigators will be notified. Since our dataset dates back to 2015, we can assume the labeling to be of very high quality.

Based on this dataset, we create datasets in the following way: We group all transactions by card holder ID and sort the transactions of each card holder by time. As a result, we obtain a temporally ordered sequence of transactions for each card holder. In the remainder of this work, we denote such sequence as the *account* of a card holder and the whole set of all accounts as the *sequence dataset*. We further split the sequence dataset into two mutually exclusive sets: One sequence dataset contains only e-commerce transactions (ECOM) and the other only face-to-face transactions (F2F).

4.1.1. Account sampling

A typical particularity of fraud detection problems is the high imbalance between the minority class (fraudulent transactions) and the majority class (genuine transactions). The overall fraction of fraudulent transactions usually amounts to about 0.5% or less. In the F2F-dataset, frauds occur less frequently by one order of magnitude as compared to the ECOM-dataset, which further aggravates the detection problem. Studies from literature (Bhattacharyya et al., 2011; Liu, Wu, & Zhou, 2009) and previous experiments showed that some form of majority class undersampling on the training set improves the learning. However, in contrast to transaction-based datasets in which transactions are considered as independent training examples, we can not directly apply such undersam-

Table 1
Dataset sizes and fraud ratios.

	Training set (01.03.–25.04.)		Validation set (26.04.–30.04.)		Test set (08.05.–31.05.)	
ECOM	$2.9 \cdot 10^6$	1.48%	$0.6 \cdot 10^6$	0.38%	$3.3 \cdot 10^6$	0.42%
F2F	$4.3 \cdot 10^6$	0.81%	$0.7 \cdot 10^6$	0.07%	$4.7 \cdot 10^6$	0.05%

pling strategy to a sequence dataset. Therefore, we employed undersampling on account level.

In this regard, an account is considered *compromised* if it contains at least one fraudulent transaction and it is considered *genuine* if it contains only genuine transactions. We employed a simple account-based sampling process to build the training set. With probability $p_g = 0.9$ we randomly picked an account from the set of genuine accounts and with probability $1 - p_g$ we picked an account from the set of compromised accounts. This process is repeated 10^6 times to create a training set with one million accounts. The de facto fraud ratio on transaction level is still smaller than 1:10 but we found that this simple approach works well in practice. See Table 1 for details regarding dataset sizes and time periods.

4.1.2. Delayed ground truth

Our test period starts more than one week after the training period. The reason is that in a production system, labels for transactions are only available after human investigators have reviewed the transactions. Therefore, the availability of an accurate ground truth is always delayed by about one week. Also, classification is typically more accurate on recent transactions that follow closely upon the training period. But such accuracy would be an overly optimistic appraisal of the classifier's performance in a production system, since in practice, we would not yet have access to the true labels.

4.1.3. Dataset alignment

Both the random forest and the LSTM were trained to predict the label of individual transactions. However, there is one difference that we need to take into account in the experiments. With an LSTM we can only predict the label of a transaction after having seen several transactions that precede it, whereas with the random forest we do not require any preceding transactions. To improve the comparability of the results, we accounted for this difference by removing all transactions that were not preceded by at least $w = 9$ preceding transactions. The RF and LSTM can now be trained, validated and tested on identical sets of transactions.

The goal with the sequence learning approach is to characterize how frauds appear in a non-fraudulent context/sequence. To study the impact of the length of the input sequence on the LSTM prediction accuracy, we defined two scenarios: In one scenario the LSTM was trained with sequences of length 5 (SHORT), in the other scenario the LSTM was trained with sequences of length 10 (LONG). The choice of these lengths is motivated from a statistical analysis in which we observed that a compromised account contains, on average, about 4 fraudulent transactions. Therefore, a sequence length of 5 transactions seemed to be an adequate lower limit to see both frauds and non-frauds within the context. Based on this lower limit, we doubled the sequence length to arrive at the second scenario with a sequence length of 10.

4.2. Features

Since the data collected during a credit-card transaction must comply with international financial reporting standards, the set of raw features is quite similar throughout the literature. Therefore, we removed all business-specific features and kept only a small set that is comparable to the features used in other studies (Bahnsen et al., 2016; Bhattacharyya et al., 2011; Carneiro et al., 2017). Also, we removed all identifiers and quasi-identifiers for card holders and merchants to encourage generalization to new frauds. Table 2 shows the list of features as published in Bahnsen et al. (2016), after removing identifiers and quasi-identifiers.

Table 2
Features in the BASE feature set (TX=transaction related, M=merchant related, CH=card holder related).

Feature name	Description	Category
Time	Date and time of the transaction	TX
Transaction type	e.g. POS, Internet, 3D Secure, PIN, ...	TX
Entry mode	e.g. magnetic stripe, contactless ...	TX
Amount	Amount of the transaction	TX
Merchant Group	Merchant group identification (MCC)	M
Country (merchant)	Country of merchant	M
Country (card holder)	Country of residence	CH
Card type	e.g. Visa debit, credit limit, expiry date	CH
Bank	Issuer bank of the card	CH

In order to assess the impact of additional features on the classification accuracy, we defined three feature sets: The first feature set (BASE) contains all raw features after removing business-specific variables and identifiers (see Table 2). The second feature set (TDELTA) contains all features from the BASE set plus the time delta feature as described in Section 3.2. The third feature set (AGG) contains all features from the TDELTA set plus 14 aggregation features as described in Section 3.2. We aggregated transactions from the past 24 h with respect to the amount spent and the number of transactions. Each aggregation feature indicates the recent activity of a card holder in a particular context. The context is defined by an element from the power set $\mathcal{P}(\{$ *merchant category code, merchant country, card entry mode*$\})$.

4.2.1. Ratio-scaled variables

We applied gaussian normalization to ratio-scaled variables such as the transaction amount or credit limit to center the variable on $\mu = 0$ with a standard deviation $\sigma = 1$. This normalization has no effect on the learning of a random forest classifier, but it accelerates the convergence of gradient-based optimization in neural networks.

4.2.2. Categorical variables

In case of the random forest classifier, categorical variables can be used just as-is. We only mapped each value to an integer number.

In case of neural networks, we wanted to avoid having very high-dimensional one-hot encoded feature vectors. Therefore, we employed a label encoding mechanism which is quite popular in the domain of natural language processing and neural networks (Collobert et al., 2011; Socher et al., 2013; Tang et al., 2014) and is applicable to arbitrary other categorical variables apart from words (Guo & Berkhahn, 2016). For a categorical variable with its set of values C, we assigned each value a random d-dimensional weight vector v, that was drawn from a multivariate uniform distribution $v \sim \mathcal{U}([-0.05, 0.05]^d)$, with $d = \lceil log_2(|C|) \rceil$. The feature values and their corresponding vectors (vector embeddings of the feature values) are stored in a dictionary. To encode a particular value of the

Table 3
Hyper-parameters considered during grid search.

RF		LSTM	
Min-sample-leaf	{1, 3, 10}	learning-rate	$\{10^{-2}, 10^{-3}, 10^{-4}\}$
Splitting-criterion	{gini, entropy}	dropout-rate	{0.2, 0.5, 0.8}
Max-features	{5, 10}	recurrent nodes (per layer)	{20, 100}
Trees	{100, 600}	hidden nodes	{20, 100}

categorical variable, we look up the value of the feature in the dictionary and retrieve its vector. The embedding vectors are part of the model's parameters and can be adjusted jointly during parameter estimation.

4.2.3. Time feature

We considered the time feature as a composition of several categorical variables. For each temporal resolution of the time feature, i.e. year, month, weekday, day, hour, minute and second, we defined a categorical variable in the same way as described above.

4.3. Classifiers

The Long Short-term memory network had two recurrent layers, a single hidden layer and a Logistic Regression classifier stacked on top of the last layer. The Logistic Regression classifier can be trained jointly with the LSTM state transition model via error back-propagation. We applied dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) to the LSTM nodes to regularize the parameters and trained the entire model by minimizing the cross-entropy between the predicted and the true class distribution with the ADAM[1] algorithm. Our implementation is based on the deep learning library Keras.[2]

Since we are studying potential benefits of an LSTM-based sequence learning approach over a static learner, we had to pick one instance from the class of static learners. Here, we chose to compare against random forests. In previous experiments we observed that random forests are a strong baseline on this task which also explains its widespread usage for fraud detection (Bahnsen et al., 2016; Carneiro et al., 2017; Ngai et al., 2011). We used the random forests implementation from SciKit-Learn (Pedregosa et al., 2011).

4.3.1. Grid search

Both the random forest and the LSTM network must be parameterized with hyper-parameters. We searched the space of possible hyper-parameter configurations in terms of a coarse grid spanned by a subset of all hyper-parameters (see Table 3). We then selected the configuration that maximized the area under the precision-recall curve up to a recall of 0.2 (Section 4.4) on the validation set.

4.4. Metrics

Two criteria guided the selection of suitable performance metrics: Robustness against imbalanced classes and attention to business-specific interests.

4.4.1. AUCPR

We employed the precision-recall (PR) curve and in particular the area under that curve to quantify the detection accuracy. The precision-recall curve is a parametric curve, meaning that each point on the PR-curve corresponds to the precision of the classifier at a specific level of recall. It is not to be confused with the ROC. ROC relates the recall of the positive class (true positive rate) with the recall of the negative class (false positive rate). In our domain the number of negative examples greatly exceeds the number of positive examples. Consequently, a large change in the number of false positives would be reflected by ROC through a very small change in the false positive rate. Such insensitivity to false positives is undesirable in fraud detection. As an alternative, precision relates the false positives to the true positives, thus capturing the effect of the large number of negative examples on the classifier's performance. For an in-depth discussion on the relationship between precision-recall and ROC curves, see the study from Davis and Goadrich (2006).

The precision-recall curve, therefore, provides a complete picture of a classifier's accuracy and it is robust even in imbalanced settings. The integral along that curve yields a single-valued summary of the performance and we denote it as AUCPR.

4.4.2. AUCPR@0.2

From a business point-of-view, low recall and high precision is preferable over high recall and low precision. A typical choice is therefore to measure the precision on the first K elements in the ranked result list. Such *precision at K* corresponds to one single point on the PR-curve but it is prone to variability due to different choices for K. In order to reflect business interests and to circumvent the variability issue, we suggest to use the integral along the PR-curve computed up to a certain recall level (0.2 in our experiments). The maximum value of *AUCPR*@0.2 is 0.2.

4.4.3. Jaccard index

To explore qualitative differences between our two approaches, we used the Jaccard Index to measure the degree to which two classifiers are similar in terms of the frauds they detect. Given two result sets (true positives) A and B, the Jaccard Index is defined as $J(A, B) = |A \cap B|/|A \cup B|$. The two sets of true positives were obtained by picking one point on the precision-recall curve - the point that corresponds to a recall of 0.2. By fixing the recall at a particular value, the number of true positives and the number of false negatives is defined implicitly. For instance, the F2F testset (see Table 1) contains 2350 frauds ($0.0005 \times 4.7 \times 10^6$). A recall, fixed at 0.2, partitions the frauds into 470 true positives and 1880 false negatives. In this analysis, we aim to quantify the agreement of a pair of classifiers in terms of the identity of their 470 true positives. We quantify this agreement by means of the Jaccard Index.

5. Results

We trained a model for each combination of feature set, data set and sequence length and then tested its classification performance on the held-out test set. In case of random forests, the length of the input sequence has no influence on the model since only the last transaction from the input sequence was used. We evaluated the trained models on each of the 24 test days individually and report their mean performance with respect to the metrics defined above.

Tables 4 and 5 show a summary of the results for the Face-To-Face and E-Commerce datasets. A first observation is that the overall detection accuracy is much higher on ECOM than on F2F,

[1] ADAM is a gradient descent optimization method with an adaptive per-parameter learning rate schedule.

[2] http://www.keras.io.

Table 4
Mean AUCPR and AUCPR$_{0.2}$ across all test days. Sequence lengths (SHORT, LONG) and feature sets (BASE, TDELTA, AGG).

		F2F			
		AUCPR (μ)		AUCPR$_{0.2}$(μ)	
	Features	RF	LSTM	RF	LSTM
SHORT	BASE	0.138	0.200	0.086	0.107
	TDELTA	0.170	0.231	0.095	0.118
	AGG	0.241	0.246	0.112	0.113
LONG	BASE	0.135	0.229	0.084	0.106
	TDELTA	0.172	0.217	0.095	0.102
	AGG	0.242	0.236	0.112	0.110

Table 5
Mean AUCPR and AUCPR$_{0.2}$ across all test days. Sequence lengths (SHORT, LONG) and feature sets (BASE, TDELTA, AGG).

		ECOM			
		AUCPR (μ)		AUCPR$_{0.2}$(μ)	
	Features	RF	LSTM	RF	LSTM
SHORT	BASE	0.179	0.180	0.102	0.099
	TDELTA	0.236	0.192	0.124	0.107
	AGG	0.394	0.380	0.158	0.157
LONG	BASE	0.179	0.178	0.101	0.104
	TDELTA	0.228	0.238	0.118	0.115
	AGG	0.404	0.402	0.158	0.160

which can be explained by the higher fraud ratio in ECOM. Secondly, longer input sequences seem to have no effect on the detection accuracy, neither for F2F nor for ECOM. Third, taking into account previous transactions with an LSTM noticeably improves fraud detection on F2F. However, such improvement is not observable on ECOM – rather are the results from the static learning and the sequence learning approach surprisingly similar.

Another observation confirms the finding that feature aggregations improve fraud detection. Their impact is much more evident on ECOM as it is on F2F. The observation that feature aggregations help in cases where the sequence model does not, suggests that these two forms of context representation are not correlated and that the approaches are complementary. Whatever information LSTM-states are tracking in the transaction history, it is not the same as what we manually added through aggregations.

The LSTM improves fraud detection on face-to-face transactions in terms of AUCPR. We were curious where this improvement is coming from. Fig. 2 displays the precision-recall curves of all model variants. In Fig. 2a we can see, that the PR-curves of RF models have a high peak in precision at low recall levels but decay quickly as the recall increases. In contrast, the LSTM models have a slightly lower precision for low recall levels but retain a higher precision as the recall increases. However, there is one interesting exception: Once we add aggregated features, the PR-curve of the random forest increases by a noticeable margin up to a performance that is on par with the LSTM models. We can not observe such gain for LSTMs. On e-commerce transactions (see Fig. 2b), the PR-curves of random forest and LSTM are almost identical across all feature sets. RF and LSTM benefit from aggregated features by the same margin.

5.1. Evolution of prediction accuracy

In Tables 4 and 5 we reported the mean statistics over all test days. When we plot the AUCPRs of RF and LSTM for individual test days, we can see in Fig. 3 that the predictions from both classifiers exhibit large variations across days. However, since the curves are correlated, we have reason to assume that on some days the detection problem is more difficult than on others. For instance, on face-to-face transactions both classifiers have their minimum with respect to the AUCPR in the time periods 9.05.–10.05. and 25.05.–26.05. Through manual inspection, we tried to link transactions from these days to calendar or public events, but we could not find an obvious answer that would explain the poor performance.

Nevertheless, the variation in the prediction accuracy on e-commerce and face-to-face transactions is not completely unexplainable. Starting from the observation that the AUCPR curves in Fig. 3 seem to follow a common trend regardless of the choice of algorithm or feature set, we conclude that the variation is an intrinsic property of the dataset. Thus, when we want to find explanatory factors for the variation, we should not look for them in the features or models but in the dataset itself. We made a first step towards such inquiry by looking at the fraud density per day (see Fig. 4). This simple statistic revealed that the fraud density can vary to a large extent from one day to another. Looking at the density curve for face-to-face, we notice that the densities from 25.5. to 31.5. can be associated with the AUCPR values on the corresponding days. Similarly, the e-commerce fraud densities in the first, middle and last test days seem to be correlated with the AUCPR values on these days (more noticeable in the AGG curve). In both scenarios, the nature of the correlation is not linear and we can always find an exception that contradicts a general statement about the direct correlation between fraud densities and prediction accuracy. However, we believe that the fraud density is, among many others, one part of the answer explaining the variation in the quality of predictions across days.

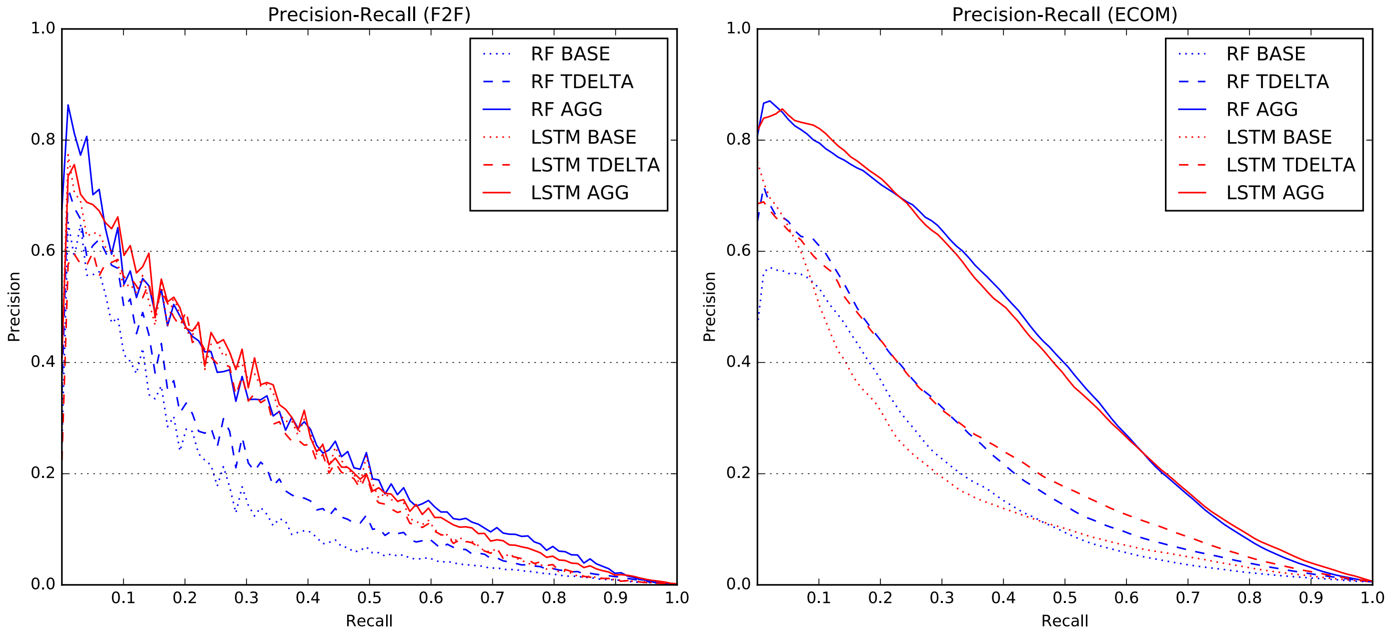
5.2. Overlapping predictions

In this analysis we had a closer look at the frauds we detected with RF and LSTM. From the set of all trained models, we picked a pair of models and compared their predictions. The decision threshold was again chosen such that it corresponded to a recall level of 0.2. All predictions with a score higher than the threshold were considered as positive predictions and all others as negative predictions. By fixing the recall, we ensured to have an equal number of true-positives in the result sets of a pair of models. However, we were interested in whether the true-positives of the RF are actually identical to the ones of the LSTM. We measured the overlap of the true-positive sets of a pair of models with the Jaccard Index. Fig. 5 displays all pairwise comparisons as a heatmap.

On both heatmaps, we observe four quite distinct areas. The two areas that correspond to intra-model comparisons and the two areas that correspond to inter-model comparisons.[3] The Jaccard Indices suggests that both the RF and LSTM are consistent in the frauds they detect. This property is slightly more pronounced in the random forest comparisons. The central and intriguing observation however, is the fact that RF and LSTM tend to detect different frauds. On F2F, RF models agree on 50.8% of their true-positives on average and LSTM models on 37.8%. Between the two model classes we observe an average agreement of only 25.2%. This is similar on ECOM with average intra-model agreements of 47.5% (RF) and 50.8% (LSTM) and an average inter-model agreement of only 35.0%.

There is one exception to this general observation. The models that were trained with aggregated features tend to detect a common unique set of frauds, that were neither detected by random forests nor LSTMs without aggregated features. This property is more pronounced on ECOM than on F2F.

[3] The Jaccard Index is symmetric and so is the matrix of all pair-wise comparisons. We kept the redundant comparisons to ease interpretation.



(a) F2F

(b) ECOM

Fig. 2. Precision-Recall curves averaged over all days in the test set (figure shows the results of the LSTM on LONG sequences).

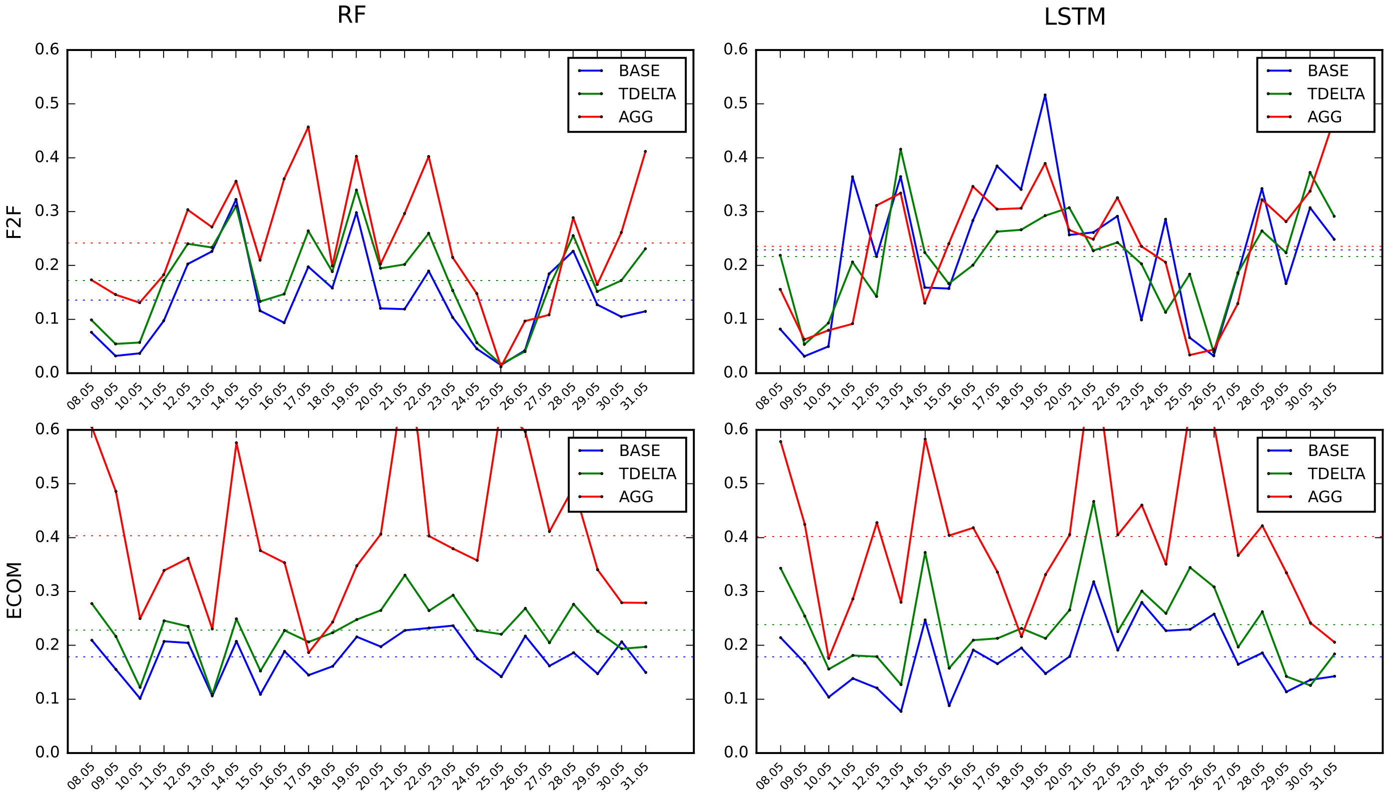


Fig. 3. Evolution of AUCPR over all test days. Horizontal dashed lines indicate the mean AUCPR for each curve (figure shows results of the LSTM on LONG sequences).

6. Discussion

6.1. Considerations for applications

During our experiments we found that the application of Long Short-Term Memory Networks to such structured data is not as straight-forward as one might think. Therefore, we would like to share some observations that might be useful for practitioners.

6.1.1. Model regularization

When we are dealing with a temporal process for which we aim to forecast some properties of future events, no collection of

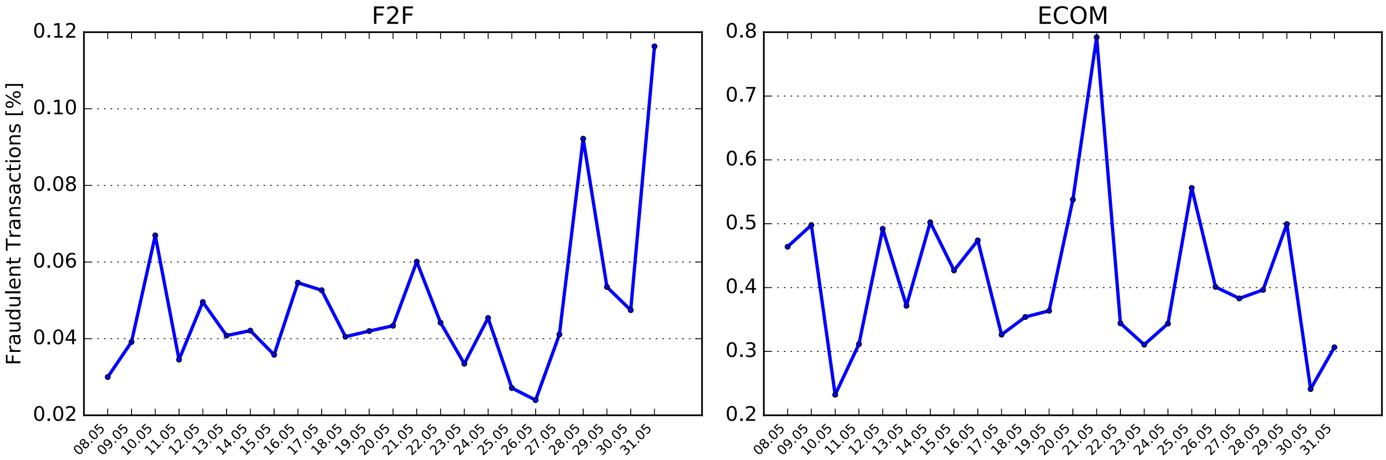
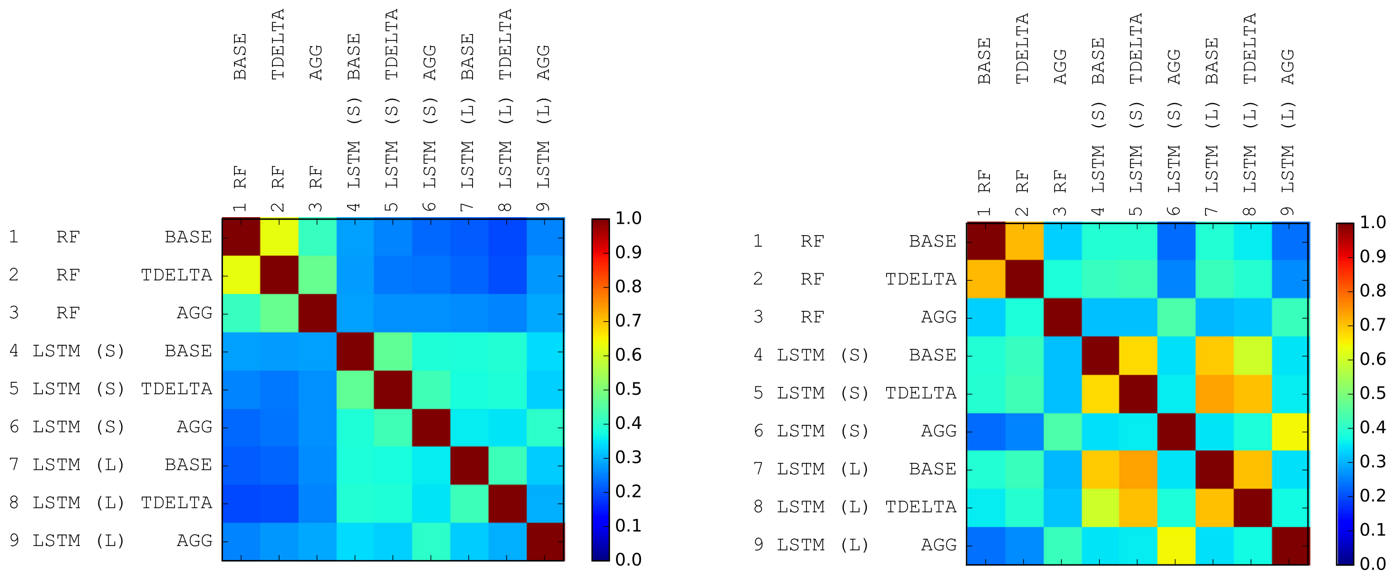


Fig. 4. Fraud density for each test day. Density is given as the percentage of fraudulent transactions in all transactions of a day.



(a) F2F

(b) ECOM

Fig. 5. Pairwise comparison of the sets of true-positives of two models measured with the Jaccard Index and color-coded in a heatmap.

historic data points can truly fulfill the requirements one would demand from a representative validation set. The prediction accuracy on days right after the end of the training set is better than on days further into the future, suggesting a time-dependency of the conditional distribution. When we choose the days right after the training period as validation set, the results on this set would suggest to apply small regularization on the model. But this choice has the contrary effect on the performance on days further into the future. An exact and highly-confident model of today's data is probably badly wrong in a few days, whereas a less confident model of today is still valid in a few days. This is less of a concern for ensemble classifiers such as random forests, but it is for Neural Networks. A neat work-around is to use *Dropout* on the network structure. Dropout samples smaller networks from the complete structure, trains these independently and finally averages the hypotheses of these smaller networks. Predictions based on this averaged hypothesis are more stable across time.

6.1.2. Online learning

Stochastic gradient descent and the many variants that have been developed for training Neural Networks (ADAM, RMSprop, Adagrad) are capable of iteratively updating the model even from imprecise errors that have been estimated on small sets of training examples. This property blends in well with the requirement of businesses to keep their detection models up-to-date with the incoming stream of transaction data.

6.1.3. Remarks on LSTM training

Due to its recurrent structure, the LSTM is prone to overfitting even when the LSTM-layers have only few nodes. Therefore, it's recommendable to start off with a rather small structure and carefully increase the size as long as there is reason to expect further generalization performance. We noticed, that a ℓ_2-penalty leads to much smoother convergence and consistently better optima than a ℓ_1-penalty. The ADAM optimizer worked much better than a vanilla SGD in our experiments since it estimates a proper learning rate schedule on the go.

6.2. Considerations for future research

Based on our study, we can immediately identify two interesting directions for future research. One is specific to credit-card fraud detection, the other is more general and might be of interest also in other domains.

6.2.1. Sequence modeling and feature aggregations

The unfortunate disadvantage of Neural Networks is the fact that they are not surrounded by a rigorous reasoning framework with clear semantics such as probabilistic models. They are powerful function approximators and we used them to approximate the conditional distribution over classes given a sequence of transactions. However, the internal mappings from a transaction to a latent state and from one state to the next are not equipped with an intrinsic interpretation due to the lack of such framework. Therefore, we can not directly query it for interesting details, i.e. finding the most likely state sequence given some transaction sequence or the most likely next transaction given the current state. Considering the results we obtained when aggregating the transaction history through manually engineered features or a sequence learner, we would need such inference mechanisms to better understand which aspects of the history we are actually aggregating with a sequence learner and why these aspects are so different from the hand-engineered features in some contexts. Quantitatively, when we add aggregated features, the sequence model becomes redundant.

6.2.2. Combined approach

Qualitatively, one difference between random forests and LSTMs remains even after adding aggregated features: the LSTM detects a different set of frauds than the random forest. Consistently more different than within the individual families. We conjecture, that this difference can be explained by the presence of sequential patterns which are guided and framed by constraints in the real-world. Therefore, the combination of a sequence learner with a static learner and aggregated features is likely to further improve the detection accuracy.

7. Conclusion

In this paper, we employed long short-term memory networks as a means to aggregate the historic purchase behavior of credit-card holders with the goal to improve fraud detection accuracy on new incoming transactions. We compared the results to a baseline classifier that is agnostic to the past. Our study showed that offline and online transactions exhibit very different qualities with respect to the sequential character of successive transactions. For offline transactions, an LSTM is an adequate model of the latent sequential patterns in that it improves the detection. As an alternative to the sequence learner, manual aggregation of the transaction history through additional features improves both the detection on offline and online transactions. However, modeling the transaction history with an LSTM yields a set of true positives that is consistently different from the frauds detected with a random forest across all feature sets which gives rise to a combined approach.

Funding

This work was supported by ATOS Worldline within the AïDA-project framework.

Acknowledgments

We want to thank ATOS Worldline for providing the financial support and the dataset that made this study possible. In particular, we want to thank Liyun He-Guelton and Olivier Caelen for their expertise in credit-card fraud detection and for pointing out the particularities of this application domain.

References

Abdallah, A., Maarof, M. A., & Zainal, A. (2016). Fraud detection system: A survey. *Journal of Network and Computer Applications, 68*, 90–113.

Bahnsen, A. C., Aouada, D., & Ottersten, B. (2015). Example-dependent cost-sensitive decision trees. *Expert Systems with Applications, 42*(19), 6609–6619.

Bahnsen, A. C., Aouada, D., Stojanovic, A., & Ottersten, B. (2016). Feature engineering strategies for credit card fraud detection. *Expert Systems with Applications, 51*, 134–142.

Bayer, J. S. (2015). *Learning sequence representations*. München, Technische Universität München, Diss. Ph.D. thesis.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks, 5*(2), 157–166.

Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. *Decision Support Systems, 50*(3), 602–613.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). Lof: identifying density-based local outliers. In *ACM sigmod record: 29* (pp. 93–104). ACM.

Carneiro, N., Figueira, G., & Costa, M. (2017). A data mining based system for credit-card fraud detection in e-tail. *Decision Support Systems, 95*, 91–101.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357.

Cieslak, D. A., Hoens, T. R., Chawla, N. V., & Kegelmeyer, W. P. (2012). Hellinger distance decision trees are robust and skew-insensitive. *Data Mining and Knowledge Discovery, 24*(1), 136–158.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research, 12*(Aug), 2493–2537.

Dal Pozzolo, A., & Bontempi, G. (2015). *Adaptive machine learning for credit card fraud detection*. Université libre de Bruxelles.

Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2018). Credit card fraud detection: a realistic modeling and a novel learning strategy. *IEEE Transactions on Neural Networks and Learning Systems, PP*(99), 1–14. doi:10.1109/TNNLS.2017.2736643.

Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd international conference on machine learning* (pp. 233–240). ACM.

Dietterich, T. (2002). Machine learning for sequential data: A review. *Structural, Syntactic, and Statistical Pattern Recognition*, 227–246.

Doya, K. (1993). Bifurcations of recurrent neural networks in gradient descent learning. *IEEE Transactions on Neural Networks, 1*, 75–80.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science, 14*(2), 179–211.

Graves, A., & Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st international conference on machine learning (ICML-14)* (pp. 1764–1772).

Graves, A. (2012). *Supervised sequence labelling with recurrent neural networks* (pp. 5–13). Berlin, Heidelberg: Springer.

Guo, C., & Berkhahn, F. (2016). Entity embeddings of categorical variables. arXiv:1604.06737.

Hemalatha, C. S., Vaidehi, V., & Lakshmi, R. (2015). Minimal infrequent pattern based approach for mining outliers in data streams. *Expert Systems with Applications, 42*(4), 1998–2012.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation, 9*(8), 1735–1780.

Jha, S., Guillen, M., & Westland, J. C. (2012). Employing transaction aggregation strategy to detect credit card fraud. *Expert Systems with Applications, 39*(16), 12650–12657.

Krivko, M. (2010). A hybrid model for plastic card fraud detection systems. *Expert Systems with Applications, 37*(8), 6070–6076.

Lafferty, J. D., McCallum, A., & Pereira, F. C. N. (2001). *Conditional random fields: Probabilistic models for segmenting and labeling sequence data* (pp. 282–289). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc..

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. In *Proceedings of the Eighth IEEE International Conference on Data Mining, 2008* (pp. 413–422). IEEE.

Liu, X.-Y., Wu, J., & Zhou, Z.-H. (2009). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39*(2), 539–550.

Mahmoudi, N., & Duman, E. (2015). Detecting credit card fraud by modified fisher discriminant analysis. *Expert Systems with Applications, 42*(5), 2510–2516.

Mikolov, T., Deoras, A., Kombrink, S., Burget, L., & Černockỳ, J. (2011). Empirical evaluation and combination of advanced language modeling techniques. In *Twelfth annual conference of the international speech communication association* (pp. 605–608).

Ngai, E., Hu, Y., Wong, Y., Chen, Y., & Sun, X. (2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems, 50*(3), 559–569.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks.. *ICML, 28*, 1310–1318.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Phua, C., Lee, V., Smith, K., & Gayler, R. (2010). A comprehensive survey of data mining-based fraud detection research. arXiv:1009.6119.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*(6088), 533.

Sahin, Y., Bulkan, S., & Duman, E. (2013). A cost-sensitive decision tree approach for fraud detection. *Expert Systems with Applications, 40*(15), 5916–5923.

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., et al. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP): 1631* (p. 1642).

Srivastava, A., Kundu, A., Sural, S., & Majumdar, A. (2008). Credit card fraud detection using hidden markov model. *IEEE Transactions on Dependable and Secure Computing, 5*(1), 37–48. doi:10.1109/TDSC.2007.70228.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research, 15*(1), 1929–1958.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104–3112).

Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., & Qin, B. (2014). Learning sentiment-specific word embedding for twitter sentiment classification.. In *ACL: 1* (pp. 1555–1565).

Van Vlasselaer, V., Bravo, C., Caelen, O., Eliassi-Rad, T., Akoglu, L., Snoeck, M., et al. (2015). Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions. *Decision Support Systems, 75*, 38–48.

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks, 1*(4), 339–356.

Whitrow, C., Hand, D. J., Juszczak, P., Weston, D., & Adams, N. M. (2009). Transaction aggregation as a strategy for credit card fraud detection. *Data Mining and Knowledge Discovery, 18*(1), 30–55.

Wiese, B., & Omlin, C. (2009). Credit card transactions, fraud detection, and machine learning: Modelling time with lstm recurrent neural networks. In *Innovations in neural information paradigms and applications* (pp. 231–268). Springer.

Zhao, X., Zhang, J., & Qin, X. (2017). Loma: A local outlier mining algorithm based on attribute relevance analysis. *Expert Systems with Applications, 84*, 272–280.