Recurrent Dynamical Projection for Time series-based Fraud detection

Eric A. Antonelo and Radu State

Interdisciplinary Centre for Security, Reliability and Trust,
University of Luxembourg,
Luxembourg

Abstract. A Reservoir Computing approach is used in this work for generating a rich nonlinear spatial feature from the dynamical projection of a limited-size input time series. The final state of the Recurrent neural network (RNN) forms the feature subsequently used as input to a regressor or classifier (such as Random Forest or Least Squares). This proposed method is used for fraud detection in the energy distribution domain, namely, detection of non-technical loss (NTL) using a real-world dataset containing only the monthly energy consumption time series of (more than 300K) users. The heterogeneity of user profiles is dealt with a clustering approach, where the cluster id is also input to the classifier. Experimental results shows that the proposed recurrent feature generator is able to extract relevant nonlinear transformations of the raw time series without a priori knowledge and perform as good as (and sometimes better than) baseline models with handcrafted features.

Keywords: recurrent neural networks, reservoir computing, non-technical loss, eletricity fraud detection, clustering, energy distribution networks.

1 Introduction

In the context of energy distribution networks, frauds are non-technical losses (NTL) that may account for up to 40% of the total energy distributed in some developing countries. Fraudsters alter (or bypass) the eletricity meter in order to pay less than the right amount. Many different methods have been used for devising fraud detection models [4, 5]. These methods usually require feature engineering on the consumptions time series data in order to train classifiers for fraud detection. Other features can also be employed such as the ones derived from customer data (e.g., spatial coordinates, neighborhood, type of residence, etc.) and textual notes written by employees of the energy distribution network responsible for reading the meters monthly - however, these notes are scarce. Whenever a note is written with respect to a customer's meter, there is a high probability of fraud. In order to verify the fraud, an inspector is sent to the field, i.e., a specialist makes a visit to the customer's residence in order to check the eletricity meter and confirm the fraud. Three outcomes are possible out of an inspection: the fraud is confirmed, there is an anomaly incurring NTL (e.g.

faulty meter or fraud not affirmed), or there was no fraud (there may also be some mislabeled data for the cases of bribery or other causes). Thus, the remaining non-inspected customers are not used in the supervised learning of the model (which can cause the so-called sample selection bias [6]). In this context, the discovery or detection of frauds is necessary to decrease the NTL of the energy distribution networks. Predictive models devised to this end have to be used with parsimony since the cost to send a inspector to confirm the fraud is expensive. Thus, only the most certain predictions (those with highest score) could be used for sending inspections, for instance. Previous work on NTL detection has employed different approaches, types of inputs and dataset sizes. For instance, [9] focuses on feature engineering with random forest, logistic regression and support vector machine as classifiers. A survey on this field is presented in [5], citing also other approaches based on fuzzy systems, genetic algorithms, etc.

This work proposes a new general purpose temporal feature extraction based on recurrent neural networks (RNNs) that projects the input stream $\mathbf{u}(t)$ into a high-dimensional dynamic space $\mathbf{x}(t)$. This projector is called *reservoir* as in Reservoir Computing (RC), whose recurrent weights are randomly initialized [11]. The states $\mathbf{x}(t)$ of the resulting dynamical system form a trajectory in the high-dimensional space that exhibits a short-term memory. This means that when a snapshot is taken from the dynamical system (i.e, at $t = t_s$), the states $x(t_s)$ sums up the recent history of the input stream. The main idea is to transform the temporal dimension of $\mathbf{u}(t)$ into a spatial dimension of the snapshoted state $x(t_s)$. In this work, this last state is used as a feature for a predictive model, which can be a regressor or a classifier. The proposed method is applied to fraud detection in the energy distribution domain where million of users with heterogeneous energy consumption profiles exist. Our work considers that the short-length time series consumption data from each user is the sole input available to the model. The proposed model also employs k-means clustering to preprocess the heterogeneous input time series as well as to provide cluster information as an additional relevant input. The resulting method is novel as far as the authors know, specially in the fraud detection domain. The experiments presented in this work show that the general-purpose recurrent feature generator achieves predictive performance at least as good as models considering handcrafted input features, and that large reservoirs and cluster information is useful mainly when using Least Squares training.

2 Methods

2.1 Recurrent dynamical projection

Our proposed method views an RNN as an automatic temporal feature generator. It transforms an input stream into a set of spatial nonlinear features that sums up the trajectory of the underlying input-driven dynamical system (see Fig. 1(a)). The RNN model we use is based on the Echo State Network (ESN) approach [7, 8]. The state update equation for the reservoir is given by:

$$\mathbf{x}(t+1) = f(\mathbf{W}_{\text{in}}\mathbf{u}(t) + \mathbf{W}_{\text{res}}\mathbf{x}(t) + \mathbf{W}_{\text{bias}}) \tag{1}$$

where: $\mathbf{u}(t)$ represents the input at time t; $\mathbf{x}(t)$ is the M-dimensional reservoir state; and $f() = \tanh()$ is the hyperbolic tangent activation function; $\mathbf{W}_{\text{in}}$ and $\mathbf{W}_{\text{bias}}$ are the weight matrices from input and bias to reservoir, respectively and $\mathbf{W}_{\text{res}}$ represents the recurrent connections between internal nodes of the reservoir. The initial state is $\mathbf{x}(0) = \mathbf{0}$. The non-trainable weights $\mathbf{W}_{\text{in}}$, $\mathbf{W}_{\text{res}}$ and $\mathbf{W}_{\text{bias}}$ are randomly generated from a Gaussian distribution $N(0, 1)$ or a uniform discrete set $\{-1, 0, 1\}$. After this random initialization, the matrix $\mathbf{W}_{\text{in}}$ ($\mathbf{W}_{\text{bias}}$) is scaled by the parameter called input scaling υ_{inp} (bias scaling υ_{bias}). Additionally, the $\mathbf{W}_{\text{res}}$ matrix is rescaled so that the reservoir has the echo state property [7], that is, the spectral radius $\rho(\mathbf{W}_{\text{res}})$ (the largest absolute eigenvalue) of the linearized system is smaller than one [7]. This means that the reservoir should have a fading memory such that if all inputs are zero, the reservoir states also approach zero within some time period. The configuration of the reservoir parameters are given in Section 3.

In this work, the reservoir is used to generate a high-dimensional feature at $t = N^{(i)}$, i.e., the reservoir states $\mathbf{x}(N^{(i)})$, where $N^{(i)}$ is the size of the i^{th} input time series $\mathbf{u}^{(i)}(t)$, which in our case is the unidimensional monthly energy consumption series. We should care that this input time series is short enough compared to the size of the reservoir such that the reservoir has enough memory to generate a dynamical state $\mathbf{x}(N^{(i)})$ summing up the main characteristics of the input stream throughout time. For each i^{th} time series, there is a feature vector $\mathbf{x}(N^{(i)})$ generated using (1) and a corresponding label $y^{(i)}$ indicating the class of the i^{th} sample - fraud (1) or non-fraud (0). The mapping $\mathbf{x}(N^{(i)}) \rightarrow y^{(i)}$ is then learned by any regression or classification algorithm (Fig. 1(b)), such as Regularized Least Squares (Ridge Regression) or Random Forest.

2.2 Clustering and Normalization

The time series data ($\mathbf{u}^{(i)}(t)$) may contain very heterogeneous streams that vary on different scales. In our current application, this means that some customers consume 1000 times more energy than others (e.g., industrial or commercial customers in relation to residential clients). Even only using normalization, the results would be sub-optimal since some samples (from high energy consumption profiles) would drive the reservoir near the saturation region of the tanh function, while others would take it to linear area around zero. We would like that the i^{th} sample $(\mathbf{u}^{(i)}(t), t = 0, \ldots, N^{(i)})$ be in a scale not very different from the rest of the samples. At the same time, we can keep the information from the original scale of the sample (consumption profile of the user) that might be lost after rescaling. Both of these things can be easily accomplished by using a clustering method such as k-means. The method work as follows: compute the mean $m^{(i)}$ of $\mathbf{u}^{(i)}(t)$; use k-means to group the samples into clusters $\beta^{(i)}$ based on the value $m^{(i)}$; rescale each sample $\mathbf{u}^{(i)}(t)$ by dividing it over the value of the center of the cluster $\beta^{(i)}$. Now, all the samples will have less disparate scales. To compensate the loss of information, we concatenate the one-hot encoding $\mathbf{c}^{(i)}$ of the cluster $\beta^{(i)}$ into the total input vector for the classifier (see Fig. 1(b)). The resulting architecture is called *Temporal Machine* (TM). A similar approach with respect

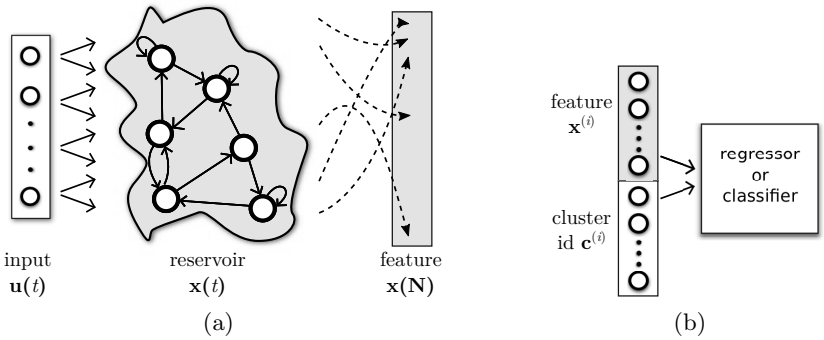


Fig. 1. (a) Recurrent Feature Generator (RFG). The reservoir is a non-linear dynamical system usually composed of recurrent sigmoid units. Solid lines represent fixed, randomly generated connections. The dashed lines show a hypothetical reservoir trajectory, ending up into a final state $\mathbf{x}(N)$ that sums up the recent history of the signal. (b) The learning machine with the temporal feature generated by RFG and the cluster id as input. We call *Temporal Machine* the conjunction of RFG + the learning module.

to using the one-hot encoding of the cluster id as input to RC networks is taken in [2]. In [1], a binary input vector is used for robot behavior learning through subspace projection in the dynamical reservoir space.

3 Experiments

3.1 Datasets

The complete dataset obtained from a certain energy distribution network in Brazil contains 3.6M customers, from which at least 800K were inspected for fraud. In this work, we only consider samples that span at least $N = 24$ months of collected energy consumption, while having a mean consumption over this time period greater than zero. These constraints reduce the dataset to $313,297$ samples, each one consisting of a time series $\mathbf{u}^{(i)}(t), t = 0, \ldots, N-1$.

3.2 Settings and Results

We made experiments using regularized Least Squares (LS) (ridge regression) [3], and random forest (RF)[1] as the regressor/classifier in the learning module of TM (Fig. 1(b)). These models are called TM-LS and TM-RF, respectively. The RF method always uses 30 trees in the forest and a maximum depth of 20. Furthermore, comparisons to baseline models are made: instead of $\mathbf{x}^{(i)}$ being generated by the RFG, it is manually computed as an 8-dimensional feature vector composed of the means and the standard deviations of the time series

[1] RF uses the method provided in the *sklearn* Python toolbox (version 0.17.1)

over the following periods: the previous 24 months, 12 months, 6 months, and 3 months. Their corresponding acronyms are LS and RF (without *TM*) for Least Squares and Random Forest, respectively. The reservoir size is $M = 100$, unless otherwise stated. Two parameters of the reservoir in the RFG are optimized with a grid search on a validation set: the spectral radius and the input scaling. This is done ten times with reservoirs whose weights are randomly initialized each time. The average AUC (area under the ROC curve) is shown in Fig. 2(a). The maximum performance is achieved for high values of the spectral radius ($\rho(\mathbf{W}_{\text{res}}) = 1$) but low values of the input scaling ($v_{\text{inp}} = 0.3$). With this optimal set of parameters, the TMs were trained using Least Squares and Random Forest on the first 80% samples and tested on the most recent 20% of the samples. The resulting ROC curves on the test set are computed using each trained model and also the baseline models LS and RF (Fig. 3). We can note that the TMs using the RFG achieves comparable performance to the baseline models with handcrafted simple features (*mean* and *std*). Additionally, the Random Forest training method seems to have better test performance than the Least Squares method. This can also be seen in Fig. 2(b), where M is varied while keeping other parameters fixed, showing that greater reservoirs matter more to Least Squares training (improving performance) than to the Random Forest method. Table 1 shows the same test AUC of the models in Fig. 3 plus two models where the reservoirs have 500 neurons (TM500-LS and TM500-RF) and other two models LS and RF without the cluster id as input. Note that the cluster id is important when LS is used, but not as much as when RF is used as training method.

Another type of evaluation was made in order to observe how the predictive model would perform on a sliding-through-time iterative train and test procedure. Fig. 4(a) shows the results of this evaluation considering the Least Squares method for training the models LS, TM-LS, and TM500-LS. Each point is one iteration of the procedure, in which a grid search on spectral radius vs input scaling is run with a validation set to find the best parameter configuration, and

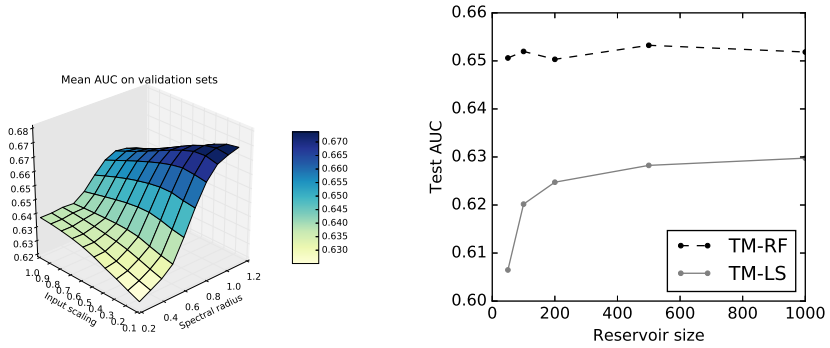


Fig. 2. (a) Grid search on spectral radius vs. input scaling showing the AUC on validation sets averaged over 10 runs each with different randomly initialized reservoirs. (b) AUC test performance vs. reservoir size for TM-RF and TM-LS in dashed and solid lines, respectively.

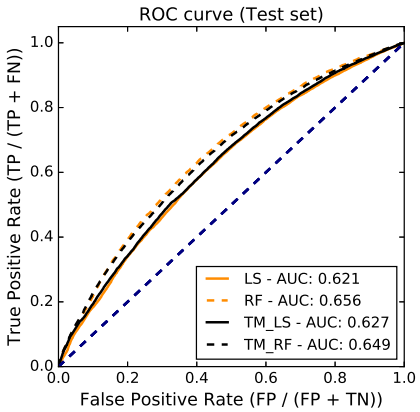


Fig. 3. (a) ROC curves on test sets (20% of the data) for 4 models: LS, RF, TM-LS and TM-RF.

then the model is evaluated on a different test set corresponding to the *current month*. The next iteration will slide the current month into the training dataset, and the new test set will be formed by samples of the following month. On average, we can see that the first two models have equal performance (the horizontal lines represent the average AUC). The latter uses a greater, more complex reservoir ($M = 500$) that can perform a little better due to its increased power for representation and temporal processing. The same procedure now with Random Forest as learning method can be visually checked in Fig. 4(b). The TM-RF model with 100 neurons in the reservoir is not as performant as the RF model. However, both of them are better then the models based on LS, on average (Table 2). The results of the sliding evaluation show that the performance is better throughout 2014 until the first quarter of 2015. In 2016, the performance drops relative to the mean AUC. One possible explanation is that detection of frauds may have explanatory variables other than the energy consumption time series not considered in this work. As a matter of comparison, [9] achieves AUC of 0.729 using random forests and handcrafted feature engineering. Their

Table 1. AUC - fixed test set

Model	Test AUC
LS (no cluster input)	0.604
RF (no cluster input)	0.654
LS	0.621
RF	0.656
TM-LS	0.627
TM-RF	0.649
TM500-LS	0.630
TM500-RF	0.653

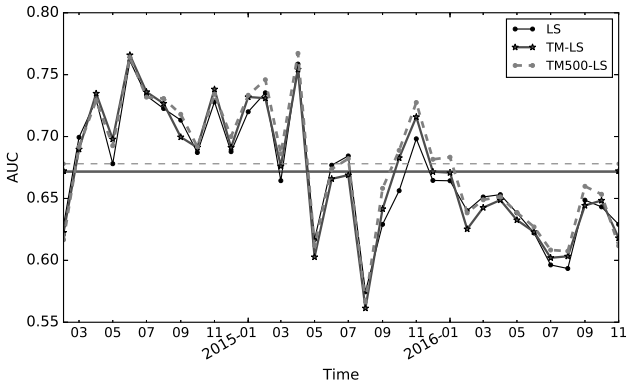
Table 2. Average test AUC over sliding evaluation

Model	Average Test AUC
LS	0.671
TM-LS	0.672
TM500-LS	0.678
RF	0.701
TM-RF	0.688

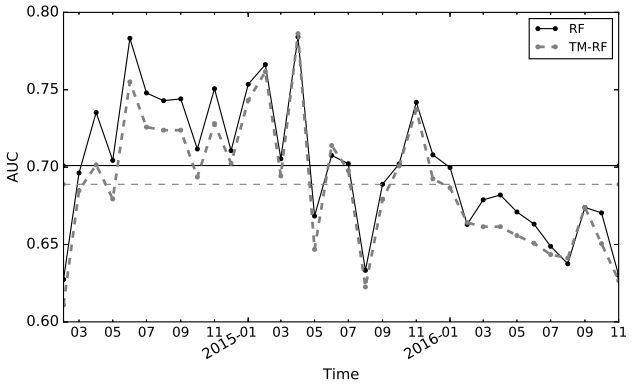
dataset is similar to the one used in this work, although being bigger, filtered and preprocessed differently, without sliding evaluations performed.

4 Conclusion

In this paper, a fraud detection method based on Reservoir Computing is proposed for detecting electricity theft having as input solely the energy consumption time series of each customer. To deal with the heterogeneous user consumption profiles, k-means clustering is employed for normalization of all the time series according to the cluster it was assigned to. The RC approach allows us to extract relevant temporal features from short-length time series by projecting the input into a high-dimensional nonlinear space. The trajectory of this input-driven dynamical system ends up into a final state which sums up the



(a)



(b)

Fig. 4. Sliding evaluation: iterative training on data starting in 2015 throughout 2016, testing each time on the following month (using AUC). Average AUC given by horizontal lines. (a) All models trained by ridge regression (regularized Least Squares estimate). (b) All models trained by Random Forest.

input time series behavior over time. The input feature used for the classifier is exactly this last state of the reservoir, showing comparable performance with baseline models using handcrafted features. The method effectively converts the temporal dimension into a spatial one, and although it was used for detecting non-technical loss in electricity grids (with real-world data) in this paper, it can also be used for general-purpose time series-based classification tasks (e.g. fraud detection). However, as the reservoir without output feedback has a limited short-term memory [7], the time series can not be indefinitely long. Furthermore, future work will research methods to optimize the reservoir dynamics (e.g., Intrinsic Plasticity [10]) in order to generate even better features. In the context of the real-world task of NTL detection, the next crucial step is to consider additional input variables such as user neighborhood, type of connection, etc. into a integrated framework which corrects the existing sample selection bias, currently considered an obstacle to the optimal use of such models.

Acknowledgments. The authors would like thank Jorge Meira and Patrick Glauner from University of Luxembourg, and Lautaro Dolberg, Yves Rangoni, Franck Bettinger and Diogo M. Duarte from Choice Technologies for useful discussions on NTL.

References

1. Antonelo, E.A., Schrauwen, B.: On learning navigation behaviors for small mobile robots with reservoir computing architectures. IEEE Transactions on Neural Networks and Learning Systems 26(4), 763–780 (2015)
2. Antonelo, E.A., Flesch, C.: Reservoir computing for detection of steady state in performance tests of compressors. Neurocomputing (accepted)
3. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer (August 2006)
4. Depuru, S.S.S.R., Wang, L., Devabhaktuni, V., Green, R.C.: High performance computing for detection of electricity theft. International Journal of Electrical Power & Energy Systems 47, 21–30 (2013)
5. Glauner, P., Meira, J., Valtchev, P., State, R., Bettinger, F.: The challenge of non-technical loss detection using artificial intelligence: A survey. International Journal of Computational Intelligence Systems (IJCIS) 10(1), 760–775 (2017)
6. Heckman, J.J.: Sample selection bias as a specification error. Econometrica 47(1), 153–161 (1979), http://www.jstor.org/stable/1912352
7. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks. Tech. Rep. GMD Report 148, German National Research Center for Information Technology (2001)
8. Jaeger, H., Haas, H.: Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication. Science 304(5667), 78–80 (Apr 2004)
9. Meira, J.A., Glauner, P., Valtchev, P., Dolberg, L., Bettinger, F., Duarte, D., et al.: Distilling provider-independent data for general detection of non-technical losses. In: Power and Energy Conference, Illinois 23-24 February 2017 (2017)
10. Schrauwen, B., Warderman, M., Verstraeten, D., Steil, J.J., Stroobandt, D.: Improving reservoirs using intrinsic plasticity. Neurocomputing 71, 1159–1171 (2008)
11. Verstraeten, D., Schrauwen, B., D'Haene, M., Stroobandt, D.: An experimental unification of reservoir computing methods. Neural Networks 20(3), 391–403 (2007)