Information Sciences 479 (2019) 448–455

Contents lists available at ScienceDirect

Information Sciences

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

Using generative adversarial networks for improving classification effectiveness in credit card fraud detection

Ugo Fiore[a,*], Alfredo De Santis[b], Francesca Perla[a], Paolo Zanetti[a], Francesco Palmieri[b]

[a] Department of Management Studies Quantitative Mathods, Parthenope University, Italy
[b] Department of Informatics, University of Salerno, Italy

ARTICLE INFO

Article history:
Received 13 August 2017
Revised 25 October 2017
Accepted 23 December 2017
Available online 25 December 2017

Keywords:
Fraud detection
Supervised classification
Deep learning
Generative adversarial networks

ABSTRACT

In the last years, the number of frauds in credit card-based online payments has grown dramatically, pushing banks and e-commerce organizations to implement automatic fraud detection systems, performing data mining on huge transaction logs. Machine learning seems to be one of the most promising solutions for spotting illicit transactions, by distinguishing fraudulent and non-fraudulent instances through the use of supervised binary classification systems properly trained from pre-screened sample datasets. However, in such a specific application domain, datasets available for training are strongly imbalanced, with the class of interest considerably less represented than the other. This significantly reduces the effectiveness of binary classifiers, undesirably biasing the results toward the prevailing class, while we are interested in the minority class. Oversampling the minority class has been adopted to alleviate this problem, but this method still has some drawbacks. Generative Adversarial Networks are general, flexible, and powerful generative deep learning models that have achieved success in producing convincingly real-looking images. We trained a GAN to output mimicked minority class examples, which were then merged with training data into an augmented training set so that the effectiveness of a classifier can be improved. Experiments show that a classifier trained on the augmented set outperforms the same classifier trained on the original data, especially as far the sensitivity is concerned, resulting in an effective fraud detection mechanism.

1. Introduction

With the astonishing growth of e-commerce over the last decade, credit cards are now the most common solution for on-line payments, opening the door to many kinds of frauds. As a consequence, implementing effective fraud detection solutions is a matter of paramount importance for all organizations issuing credit cards or managing online transactions, in order to reduce losses and simultaneously improve customers' confidence. State-of-the-art fraud detection systems aim at identifying suspicious usage patterns from transaction logs, where illicit transactions are mixed with legitimate ones, by using sophisticated analytics and data mining techniques. This implies gaining insights into very large datasets and performing binary classification in order to discriminate fraudulent or anomalous transactions from legitimate ones. Machine

* Corresponding author.
E-mail address: ufiore@unina.it (U. Fiore).

https://doi.org/10.1016/j.ins.2017.12.030
0020-0255/

learning demonstrated to be extremely effective for facing this challenge, and in particular supervised classification techniques, where pre-classified datasets containing labeled transactions are used for training a classifier that builds a detection model capable to spot anomalous transactions among normal ones.

Unfortunately, due to the extremely small number of illicit transaction records typically available over the total ones, the supervised classification approach is known to be adversely affected by the class imbalance problem. When training data are imbalanced, some classes (in our binary classification case, the one associated to legitimate transactions) are proportionally more represented than others. On the contrary, the number of examples for a rare class may also be so low in a dataset that a learning algorithm might discard such examples, treating them as noise and classifying all examples as instances of the majority class [23]. Machine learning algorithms typically aim at maximizing accuracy—which implicitly means that all misclassification errors are treated equally—and therefore do not respond well to imbalanced datasets [20]. Indeed, it is a well known fact in machine learning that classification results are undesirably biased toward the prevailing class in the training dataset. Put differently, for well-represented classes in training data, classification errors tend to be lower than in the case of classes with few instances. However, class frequencies in training data not always reflect the prior probabilities of classes.

In many practical binary classification problems – support to medical diagnosis, network intrusion prevention, or fraud detection, to name only a few – the different misclassification errors (type I or type II – respectively false positives or false negatives) can have widely different costs (or benefits, as suggested in [13]) and it might be essential to control, to a certain degree, the compromise between those errors. For example, in the Neyman-Pearson (NP) framework [32], the maximum tolerable false positives rate (FPR) is set at a specified value α, and the goal is to minimize the false negative rate (FNR) subject to the condition that FPR is no greater than α. This arises quite naturally in many settings, especially when the class of interest is the minority one, as in our specific case.

Our objective is to generate a large number of convincing (and reliable) examples of the minority class that can be used to re-balance the training sets used by the binary classifier. This requires a deep understanding of the different dynamics of fraudulent and legitimate transactions, by an analysis on available data.

Generative Adversarial Networks (GANs) are deep learning technologies, building up multiple layers of abstraction to learn hierarchies of concepts, that have achieved considerable success in generating convincing examples, especially real-looking images. They are very general, though, and can be applied to several settings. GANs are composed of two models, a generative one and a discriminative one, which compete against one another, playing a zero-sum minimax game [17]. In the usual setting, where the two adversaries are multilayer perceptrons, some of the issues arising with training GANs are reduced.

To address the imbalanced dataset problems in supervised classification-based credit card fraud detection, we build an *augmented* training set where the number of "interesting but underrepresented" instances is higher than in the original training set. To this end, we generate a collection of *credible* examples by means of a GAN which has been exposed to the original "interesting" instances extracted from the training set. The GAN has been trained to mimick the original minority class examples as closely as possible. By the nature of the GAN, synthetic examples will be indistinguishable from original ones, at least from the point of view of the discriminative component of the GAN. We merged synthetic examples with original training data, obtaining an augmented training set that is more balanced, so that the desired effectiveness can be achieved by using a traditional classifier. A careful experimental evaluation showed that a classifier trained on the augmented set largely outperforms the same classifier trained on the original data, especially as far as sensitivity is concerned, resulting in a really effective credit card fraud detection solution. While our framework is presented here in the context of credit card fraud detection, it should be remarked that is quite general and it can readily be extended to other application domains.

The next Section is dedicated to a review of related work. A brief summary of GANs is provided in Section 3. Our approach, based on GANs, to achieve an improved discriminating ability in the context of credit card fraud detection is presented in Section 4, and validated experimentally in Section 5. Finally, Section 6 concludes the paper.

2. Related work

Transactional fraud detection based on data mining, mainly related to credit card usage, has received a great attention from the research world, since many data warehouses are making available large volumes of data that can be carefully analyzed. Supervised approaches based on machine learning are now considered the most promising and effective solutions available. One of the first experiences in credit card fraud detection using several data-visualization methods with supervised learning has been proposed in [3]. A predictive supervised solution examining labeled transactions to determine what typical fraudulent transactions look like has been presented in [34]. Neural networks have been used for detecting credit card frauds through massive data mining in [7]. A neural network based on a three-layer, feed-forward Radial Basis Function has been used to assign a fraud score to new credit card transactions in [15], whereas another supervised approach for customer-specific credit card fraud detection based on fuzzy neural networks has been presented by Syeda et al. [35]. A supervised solution based on SVM with bagging and boosting has been proposed for detecting telecommunication subscription frauds in [25]. A comparison between supervised fraud detection strategies based on neural and bayesian networks has been presented in [29], whereas a boosted naive bayes classifier has been used in [36]. Decision tree classifiers have been used in rule-based fraud detection solutions, such as the ones presented in [31] and [33], using modified C4.5 algorithms.

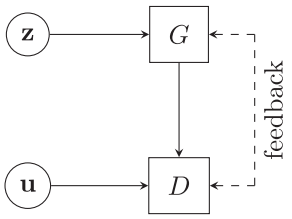

Fig. 1. The generator G receives random noise $\mathbf{z}$ as input and its output is given to the discriminator D, whose objective is to distinguish the examples produced by G from original data $\mathbf{u}$.

The technical problems associated to class imbalance in fraud detection, where only a minority of training instances are of interest, has been discussed in [24].

The general imbalance problem has been dealt with by devising modified classifiers or by preprocessing data before any classification algorithm is applied [28]. Davenport [11] evaluated two approaches for adjusting the false positive and false negative rates of an SVM (Support Vector Machine) classifier: shifting an offset parameter, resulting in an affine shift of the decision boundary, and introducing an additional parameter to control the relative weight given to each class. The same authors remarked, in a subsequent paper, that it is extremely important to have accurate estimates of the FPR and the FNR [12]. An extension of the hinge loss function for an SVM classifier, with interesting connections to risk minimization, was proposed in [30]. Ensemble methods have also been applied to imbalanced datasets [14]. These methods work by training a number of classifiers and combining their output to produce a single decision.

A strategy for alleviating the imbalanced-class problem is to reduce the disparity between classes by undersampling the majority class in a training set [1], randomly removing some instances of the majority class, on the assumption that redundancy in data makes such removal irrelevant for classification purposes. Dal Pozzolo et al. [10] investigated the bias arising from undersampling. Oversampling the minority class has been also attempted [20] by replicating instances of the minority class. A drawback of such strategy is that does not add informative content, thus limiting the improvement on the ability of a classifier to generalize. To partially overcome this difficulty, the minority class is oversampled in the Synthetic Minority Oversampling Technique (SMOTE) [9], where synthetic examples are generated by interpolating between examples of the same class. This has the effect of creating clusters around each minority observation. SMOTE has given rise to several variants, notably Border SMOTE [19] which concentrates synthetic examples along the borderline between classes, and DBSMOTE [8], a cluster-based algorithm relying on an organization of data into clusters by means of DBSCAN clustering.

On the contrary, we seek the same goal, i.e., rebalancing a training set, by means of injecting credible examples for the minority class. In this way, no straightforward oversampling is performed on the minority class. Rather, the task of generalizing from examples of the minority class alone is delegated to a GAN, which has shown satisfactory performance in generating credible examples. In addition, no information is taken out from the training set.

3. Generative adversarial networks

A GAN consists of two feed-forward neural networks, a Generator G and a Discriminator D competing against each other, with G producing new candidates and its adversary D evaluating their quality. Each of the two networks is usually a deep neural network [27], with several layers connected in such as way that the output of the units in each layer becomes the input for the units in the layer immediately above. Viewing what is learned at each layer as a representation of the original input, layers can be ideally associated to levels of abstraction or composition capabilities. Changing the number of layers and the layer size allows to achieve varying degrees of abstraction [5]. The farthest a layer is from the original input, the higher is the level of abstraction of its representation, because higher-level features are defined in terms of lower-level features. Deep networks are thus capable of discovering rich hierarchical models by exploiting the aforementioned concept of hierarchical explanatory schemes, where at higher levels more abstract concepts are learned [22]. Such mechanisms have shown to be apt to richly describe data in several real-world application domains.

The main idea in GANs is to refine a generative model by making it confront an adversary, a discriminative model that has the goal of separating the generated examples from real ones. The generator takes random noise $\mathbf{z}$ as input, transforms it through a function and produces examples, while the discriminator learns to determine whether an example has been produced by the generator (see Fig. 1). That is, the generator network has the purpose of learning the probability distribution of training data, by mapping $\mathbf{z}$ to such distribution, in order to produce new artificial candidates that are as close as possible to real data instances. On the other hand, the goal of the adversarial discriminator network is to differentiate correctly between real data and artificial candidates, by penalizing the generator's activity of producing artificial candidate instances. Clearly, the generator network struggles to cheat the discriminator network by producing synthesized instances that appear to be as realistic as possible, in order to increase the error rate of its adversary, resulting in a cat-and-mouse game where both competitors improve their ability until an equilibrium is reached, where generated examples are indistinguishable from real ones. Thus, in such a competitive inferential setting, the generator network produces more and more realistic instances

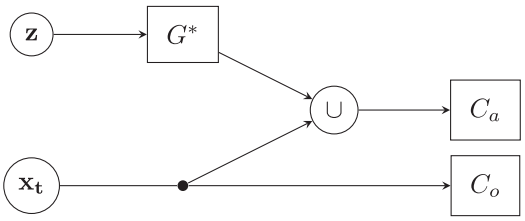

Fig. 2. The trained generator G^* is fed with random noise $\mathbf{z}$ and its output is merged with the original training set $\mathbf{x_t}$. The same classifier C is trained on the augmented (C_a) and the original training set (C_o).

over time, whereas the discriminator continuously improves its capability of distinguishing real instances from synthesized ones.

The training goal for the generator is tricking the discriminator into believing that generated examples are real. The discriminator is trained by minimizing its prediction error, whereas the generator is trained on the basis of maximizing the prediction error by the discriminator. This results in a competition between generator and discriminator that can be formalized as a minimax game:

$$\min_{\theta_G} \max_{\theta_D} \left(\mathbb{E}_{\mathbf{x}\sim p_{\mathcal{D}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathcal{Z}}}[\log\left(1 - D(G(\mathbf{z}))\right)]\right)$$

where $p_{\mathcal{D}}$ is the data distribution, $p_{\mathcal{Z}}$ is the prior distribution of the generative network, and $\boldsymbol{\theta}_G$ (resp., $\boldsymbol{\theta}_D$) are the parameters of the generator (resp., discriminator) network. In other words, the goal of generator is keeping the differentiation between real and generated data to a minimum, whereas the discriminator aims to maximize the probability of distinguishing real data from generated ones.

Training GANs is known to not be an easy task [2]. The limited modeling capability of the generator can prevent it from being able to reproduce all nuances of the data. The most critical issue related to GAN training is stability, or, in other words, balancing between its component networks. When training a GAN, if the discriminator turns out to be significantly more effective than its generative counterpart, the entire GAN would not be correctly trained. Early on in training, the discriminator gets better very quickly and it is difficult for the generator to match the improvements of its rival. A similar effect will be observed in presence of a discriminator that becomes too weak with respect to its rival generator, also resulting into a useless setting. We can also consider the GAN stability into a holistic convergence perspective. Both component networks compete in order to prevail on the other one, so that they strongly depend on each other for an effective training. In presence of a severe unbalancing where a component fails against the other, the whole GAN fails.

4. The credit card fraud detection framework

Fraudsters continually target credit card payment systems and enterprises need to monitor transactions, detecting and preventing fraudulent behavior in order to preserve customer trust in electronic payment systems [6]. Credit card fraud detection can be formulated as a binary classification task where a vector of features and a class is associated to each transaction record. Typically, credit card fraud datasets are severely imbalanced, because fraudulent transactions are only a small fraction of non-fraudulent ones. The class of interest is the minority class.

Our framework can be outlined as follows.

(i) Train a classifier on the original data, identifying the hyperparameters for the classifier C providing the best perfomance on the testing set.
(ii) Isolate all fraudulent examples in the training set T. Denote the resulting set by F
(iii) Use set F as training set for a GAN, tuning its hyperparameters
(iv) Use the trained generator, name it G^*, of the GAN to generate synthetic examples F', receving as input random noise $\mathbf{z}$
(v) Merge F' into the training set T and compare the performance, on the same testing set, of C trained on the augmented set (C_a) with the performance of the same discriminator trained on the original training set (C_o)

Note that in our framework, two very different discriminators operate. The first is the discriminator within the GAN, whose purpose is to distinguish the synthetic examples produced by the GAN generator G from the real examples; the second is the classifier C whose task is to identify fraudulent examples from non-fraudulent ones. To visually emphasize that distinction, we will use the term *discriminator* and the notation D for the GAN component and the term *classifier* and the notation C for the algorithm in the final stage. When it is necessary to clarify the difference between the classifiers trained on the original and on the augmented training set, we will use C_o for the former and C_a for the latter. In the experiments, we used for C a deep neural network which showed remarkable performance. Note that any other discriminator could have been used instead.

5. Experiments

Obtaining credit card fraud datasets is difficult, because banks are reluctant to make such data public. Our experiments were performed on a publicly available dataset, the Credit-card dataset [10], containing 284,807 credit card transactions made over two days in September 2013 by european cardholders. The 492 positive class examples (i.e., fraudulent transactions) account for 0.172% of all transactions in that dataset. Due to a confidentiality request by the institution releasing the data, the Credit-card dataset contains numerical features, labeled `V1` to `V28`, which are the principal components resulting from Principal Components Analysis applied to the original features, with the exception of the first feature, `Time`, the time in seconds between transactions, and the last two, `Amount`, the involved amount, and `Class`, the response variable taking value 1 for frauds and 0 otherwise. We further preprocessed the Credit-card dataset to remove duplicates and to rescale the features in the interval [0, 1]. The resulting dataset contained 446 fraudulent transactions out of 283,726. The dataset was then partitioned in a training set, accounting for two thirds of the data, and a testing set comprising the remaining third. Thus, we have 315 fraudulent records (out of 170,236, for an incidence of 0.185%) in the training set and the remaining 131 (out of 113,490, for an incidence of 0.115%) in the testing set.

Tests were performed with the `pylearn2` research library [18]. Many classifiers based on deep neural networks have been tested, selecting for C the one that achieved the best performance. As is generally the case with deep neural networks, several hyperparameters influence the system behavior. Is it essential that such hyperparameters are selected and tuned with care. To start with, the number of layer in a deep network is a fundamental hyperparameter. Too few layers will hinder the ability of a network to build a representation at a level of abstraction which is appropriate to adequately capture data complexity, too many layers will complicate training substantially and will likely cause overfitting. As a reasonable tradeoff, networks with 2 and 3 layers were tested in the generator, in the discriminator, and in the classifier.

In each layer, units operate a transformation of their input, typically a nonlinear *activation function* applied (componentwise) to a linear combination $\mathbf{Wu} + \mathbf{b}$, where $\mathbf{u}$ is the vector of inputs to the unit, $\mathbf{W}$ is a matrix of weights and $\mathbf{b}$ is a vector of additive biases. The number of units need not be the same for each layer. Values tested for the number of units were in the range from 20 to 40, with the exception of the first layer in the generator G, which seemed to require a larger number of units (around 100). Such values were found empirically, observing that performance dropped significantly upon approaching the extreme values in the ranges detailed above. Within the ranges, instead, variation in performance was less dramatic, with minor fluctuations. As regards the type of activation function in each layer, alternatives explored included the logistic sigmoid, defined as

$$\mathrm{sigm}(\mathbf{v}) = \frac{1}{1 + e^{-\mathbf{v}}}$$

Rectified Linear Units (ReLu) [16], defined as

$$\mathrm{rectifier}(\mathbf{v}) = 0 \vee \mathbf{v}$$

and the hyperbolic tangent tanh. Weights and biases are adapted to the input on the basis of the desired output by means of training, when such parameters are progressively modified (on the basis of the learning rate) so that performance improves. A bad choice of initial values for biases and weights may have an adverse effect on the training algorithm, causing a degradation in performance that can even lead to the exhaustion of the computing time budget. In fact, as a protection against excessively long computation, training is usually stopped when a predetermined number of iterations is reached or when no significant improvement can be made on an iteration. In the latter case, a grace period can also be set, so that an additional (small) number of iterations is allowed before stopping, to handle transient plateaus in the objective function. For the initialization of weights and biases, values drawn from a normal $\mathcal{N}(0, 1)$ distribution and a uniform distribution $\mathcal{U}(-0.5, +0.5)$ were tested. Learning rates in a logarithmic grid $(5 \times 10^{-4}, 5 \times 10^{-3}, 5 \times 10^{-2})$ were experimented. Momentum [21] was initially set to 0.5, and linearly increased during the first 10 training epochs to reach a 0.99 saturation level. Simplified Nesterov momentum as described in [4, Section 3.5] was also adopted.

Metrics used in the experiments included the sensitivity (commonly known as recall in machine learning applications), the specificity, the precision (also referred to as positive predictive value), the F-measure, and the accuracy. The *sensitivity* quantifies the ability to detect a fraudulent example when it is actually fraudulent, the *specificity* is the proportion of actually non-fraudulent examples that are correctly recognized as such, while the *precision* is the proportion of predicted positive examples that are actually positive. The F-measure is the harmonic mean of precision and recall, and the *accuracy* is the proportion of predictions that are correct. We reported all these values since it is widely agreed that the accuracy alone is unable to provide an accurate description of the performance of a classifier acting on imbalanced datasets, and sensitivity and specificity have been criticized in that context as well. It is worth to remind that in applications such as credit card fraud detection, the cost of a false positive and of a false negative are not equal. An ideal fraud detection system should identify precisely the fraudulent transactions, preventing financial loss, while at the same time reducing the number of false positives that require control by human investigators, with significant costs.

The best performing classifier was found to be a 3-layer network composed of 30 ReLu units, 30 Sigmoid units, and 2 Softmax units for outputting the final class, obtaining the results shown in the first row of Tables 1 and 2.

It should be noted that our choice of training the GAN with only the “interesting” (in this application, fraudulent) examples has an important side-effect. Since training and tuning a GAN is an expensive operation, having it work on a set with low cardinality enables the achievement of excellent performance in what could have been the bottleneck of our system.

Table 1
Sensitivity and specificity as the number N_g of generated examples is varied.

N_g	Sensitivity		Specificity	
	GAN	SMOTE	GAN	SMOTE
0	0.70229	0.70229	0.99998	0.99998
79	0.70229	0.71247	0.99997	0.99997
158	0.71756	0.70229	0.99994	0.99998
315	0.72519	0.72519	0.99992	0.99994
630	0.73282	0.69466	0.99994	0.99997
945	0.72519	0.70229	0.99994	0.99996
1260	0.72519	0.70229	0.99994	0.99998
2520	0.72519	0.70229	0.99994	0.99998
3150	0.73028	0.71756	0.99994	0.99996
6300	0.72519	0.70229	0.99995	0.99998
31500	0.70229	0.70229	0.99996	0.99998

Table 2
Precision, *F*-measure, and accuracy as the number N_g of generated examples is varied.

N_g	Precision		*F*-measure		Accuracy	
	GAN	SMOTE	GAN	SMOTE	GAN	SMOTE
0	0.97872	0.97872	0.81778	0.81778	0.99964	0.99964
79	0.96842	0.96552	0.81416	0.81991	0.99963	0.99964
158	0.93069	0.97872	0.81034	0.81778	0.99961	0.99964
315	0.91346	0.93137	0.80851	0.81545	0.99960	0.99962
630	0.93204	0.96809	0.82051	0.80889	0.99963	0.99962
945	0.93137	0.95833	0.81545	0.81057	0.99962	0.99962
1260	0.93137	0.97872	0.81545	0.81778	0.99962	0.99964
2520	0.93137	0.97872	0.81545	0.81778	0.99962	0.99964
3150	0.93182	0.94949	0.81883	0.81739	0.99963	0.99963
6300	0.94059	0.97872	0.81897	0.81778	0.99963	0.99964
31500	0.95833	0.97872	0.81057	0.81778	0.99962	0.99964

Hyperparameters for the GAN were empirically determined similarly to what had been done for the hyperparameters of the classifier. The discriminator D is a 3-layer perceptron with each of the hidden layers composed of 36 Sigmoid units. The generator G also has three layers, with ReLu units in the first two and Sigmoid units in the third one. The dimension of the noise vector $\mathbf{z}$ was set to 100.

The number N_g of generated examples can be chosen at will, because such examples will be produced very efficiently by feeding the trained generator G^* in the GAN with random noise. It is, then, interesting to study how the performance of our fraud detection system varies when a different number of generated examples are injected into the augmented training set. We tested amounts of N_g in a grid with values equal to 1/4, 1/2, 1, 2, 3, 4, 8, 10, 20, 100 times N_t, the number of minority class examples in the original training set (see Fig. 1).

As it can be seen from Table 1, classification sensitivity improves appreciably when an augmented training set is used. This is counterbalanced by a moderate increase in false positives, corresponding to a slight decrease in specificity. That could be expected, since the augmented training set may include spurious data that are unrelated to the "true" essence of fraudulent transactions but that the discriminator in the GAN failed to identify. In particular, an injection of twice as much synthetic fraudulent examples as there are original fraudulent ones is the best compromise, as the *F*-measure confirms (see Table 2).

A comparison with SMOTE in its plain version was also carried out. Such comparison is significant because SMOTE, similarly to our framework, operates on the minority examples only. The results for SMOTE are also reported in Tables 1 and 2. It can be seen that the sensitivity of SMOTE is generally smaller, while the specificity tends to be higher in comparison to our framework. The *F*-measure values for SMOTE show a limited variation with respect to the number of synthetic examples and an overall better performance of our framework, especially for values in the middle of the table.

Finally, we remark that an ensemble method [26] could be used to combine the improved sensitivity of the classifier C_a trained on the augmented dataset with the high specificity of the classifier C_o trained on the original dataset.

6. Conclusions

In this work, a strategy is presented to deal with the problem of class imbalance in the application of supervised classification to detection of credit card fraud. Given a training set, an augmented set is produced, containing more examples

of the minority class with respect to the original. Synthetic examples are generated by means of a tuned GAN. This means that the discriminator component of the GAN is unable to separate synthetic examples from real ones.

While we have presented our framework in the context of credit card fraud detection, it is quite general and can readily be extended to other application domains characterized by significant class imbalance rates. We are actively pursuing that endeavor. We have seen how, as it could be expected, injecting synthetic examples in a training set causes an increase in false positives. Clearly, this can be a limiting factor in settings where the cost of a false positive is relatively high. However, the use of ensemble methods can remedy that and it is worth investigating. The proposed mechanism is intrinsically dependent on the availability of labeled instances of fraudulent transactions. In an unsupervised setting, it would be difficult to apply our framework. It should be taken into account that, in the specific context of credit card fraud, customer complaints are a valuable source of labeled data. Finally, while our method can handle frauds which are similar, in essence, to malevolent transactions seen before, it can be expected to be largely ineffective in spotting frauds that are completely novel, where there is no information to generalize upon.

To validate our framework, we performed experiments on publicly available credit card fraud detection data, where the minority class is severely underrepresented. Our framework achieved an improved sensitivity at the cost of a limited increase in false positives. Another interesting point is computing performance, since the potentially costly portion of the elaboration, training the GAN, is done on a small subset of the training data. Directions for future study are manifold. We are planning to devise a strategy to reduce the decrease in specificity to a minimum.

References

[1] R. Akbani, S. Kwek, N. Japkowicz, Applying support vector machines to imbalanced datasets, in: In Proceedings of the 15th European Conference on Machine Learning (ECML, 2004, pp. 39–50.
[2] M. Arjovsky, L. Bottou, Towards principled methods for training generative adversarial networks, in: Proceedings of the 5th International Conference on Learning Representations, 2017.
[3] B.G. Becker, Using mineset for knowledge discovery, IEEE Comput. Graph Appl. 17 (4) (1997) 75–78.
[4] Y. Bengio, N. Boulanger-Lewandowski, R. Pascanu, Advances in optimizing recurrent networks, in: Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, IEEE, 2013, pp. 8624–8628.
[5] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.
[6] S. Bhattacharyya, S. Jha, K. Tharakunnel, J.C. Westland, Data mining for credit card fraud: a comparative study, Dec. Support Syst. 50 (3) (2011) 602–613.
[7] R.J. Brachman, T. Khabaza, W. Kloesgen, G. Piatetsky-Shapiro, E. Simoudis, Mining business databases, Commun. ACM 39 (11) (1996) 42–48.
[8] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, Dbsmote: density-based synthetic minority over-sampling technique, Appl. Intell. 36 (3) (2012) 664–684.
[9] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, Smote: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2002) 321–357.
[10] A. Dal Pozzolo, O. Caelen, R.A. Johnson, G. Bontempi, Calibrating probability with undersampling for unbalanced classification, in: IEEE Symposium on Computational Intelligence and Data Mining, IEEE, 2015, pp. 159–166.
[11] M.A. Davenport, R.G. Baraniuk, C.D. Scott, Minimax support vector machines, in: IEEE/SP 14th Workshop on Statistical Signal Processing (SSP '07), 2007, pp. 630–634.
[12] M.A. Davenport, R.G. Baraniuk, C.D. Scott, Tuning support vector machines for minimax and Neyman–Pearson classification, IEEE Trans. Pattern Anal. Mach. Intell. 32 (10) (2010) 1888–1898.
[13] C. Elkan, The foundations of cost-sensitive learning, in: Proceedings of the 17th International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers Inc., 1, 2001, pp. 973–978.
[14] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, F. Herrera, A review on ensembles for the class imbalance problem: bagging–, boosting–, and hybrid–based approaches, IEEE Trans. Syst. Man Cybern. Part C 42 (4) (2012) 463–484.
[15] S. Ghosh, D.L. Reilly, Credit card fraud detection with a neural-network, in: System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on, IEEE, 3, 1994, pp. 621–630.
[16] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323.
[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Proceedings of the 27th International Conference on Neural Information Processing Systems, 2014, pp. 2672–2680.
[18] I.J. Goodfellow, D. Warde-Farley, P. Lamblin, V. Dumoulin, M. Mirza, R. Pascanu, J. Bergstra, F. Bastien, Y. Bengio, Pylearn2: a machine learning research library, arXiv:1308.4214 (2013).
[19] H. Han, W.-Y. Wang, B.-H. Mao, Borderline-smote: a new over-sampling method in imbalanced data sets learning, Adv. Intell. Comput. (2005) 878–887.
[20] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (9) (2009) 1263–1284.
[21] G. Hinton, A practical guide to training restricted boltzmann machines, Momentum 9 (1) (2010) 926.
[22] A.G. Ivakhnenko, Polynomial theory of complex systems, IEEE Trans. Syst. Man Cybern. 1 (4) (1971) 364–378.
[23] N. Japkowicz, S. Stephen, The class imbalance problem: a systematic study, Intell. Data Anal. 6 (5) (2002) 429–449.
[24] D. Jensen, Prospective assessment of ai technologies for fraud detection: a case study, in: AAAI Workshop on AI Approaches to Fraud Detection and Risk Management, 1997, pp. 34–38.
[25] H.-C. Kim, S. Pang, H.-M. Je, D. Kim, S.Y. Bang, Constructing support vector machine ensemble, Pattern Recognit 36 (12) (2003) 2757–2767.
[26] L.I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, 2nd ed., John Wiley & Sons, 2014.
[27] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[28] V. López, A. Fernández, S. García, V. Palade, F. Herrera, An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics, Inf. Sci. 250 (2013) 113–141.
[29] S. Maes, K. Tuyls, B. Vanschoenwinkel, B. Manderick, Credit card fraud detection using Bayesian and neural networks, in: Proceedings of the 1st International Naiso Congress on Neuro Fuzzy Technologies, 2002, pp. 261–270.
[30] H. Masnadi-Shirazi, N. Vasconcelos, Risk minimization, probability elicitation, and cost-sensitive SVMs., in: 27th International Conference on Machine Learning (ICML '10), 2010, pp. 759–766.
[31] S. Rosset, U. Murad, E. Neumann, Y. Idan, G. Pinkas, Discovery of fraud rules for telecommunications?challenges and solutions, in: Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 1999, pp. 409–413.
[32] C. Scott, R. Nowak, A Neyman–Pearson approach to statistical learning, IEEE Trans. Inf. Theory 51 (11) (2005) 3806–3819.
[33] H. Shao, H. Zhao, G.-R. Chang, Applying data mining to detect fraud behavior in customs declaration, in: Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on, IEEE, 3, 2002, pp. 1241–1244.
[34] E. Sherman, Fighting web fraud., Newsweek 139 (23) (2002). 32B–32B

[35] M. Syeda, Y.-Q. Zhang, Y. Pan, Parallel granular neural networks for fast credit card fraud detection, in: Fuzzy Systems, 2002. FUZZ-IEEE'02. Proceedings of the 2002 IEEE International Conference on, IEEE, 1, 2002, pp. 572–577.
[36] S. Viaene, R.A. Derrig, G. Dedene, A case study of applying boosting naive Bayes to claim fraud diagnosis, IEEE Trans. Knowl. Data Eng. 16 (5) (2004) 612–620.