COMPUTER SCIENCE MASTER DEGREE
UNIVERSITY OF BOLOGNA

Autoencoders for credit card fraud detection

Gianluca Principini
January 25, 2018

INTRODUCTION

A fraud is defined by Cambridge Dictionary as "the crime of getting money by deceiving people". This represents a billion dollar business which increases every year and evolves with technologies.

Fraud detection techniques fall into two primary classes: statistical techniques and AI techniques. Those include expert systems, data mining, pattern recognition and machine learning. In particular, traditional approaches have been rule-based expert systems. The main problem concerned the flexibility, because it wasn't hard for criminals to bypass these rules. Nowadays it is possible to exploit Big Data, using Data Science and Machine Learning techniques. In machine learning for fraud detection literature we find both supervised techniques that make use of the class of the transaction and unsupervised techniques. The first ones require to be confident on the data used to build the model. In contrast, unsupervised methods simply seek those accounts, customers and so forth which are most dissimilar from the norm. Outliers are a basic form of nonstandard observation. These methods are capable of detecting new fraudolent behaviours [2] [5]

Fraud detection datasets are usually unbalanced and features are not labeled due to privacy issues. Learning from unbalanced datesets is difficult because traditional supervised learning systems are not designed to cope with a large difference between the number of instances for each class. They rely on sampling techniques to balance the dataset. [3] Therefore unsupervised learning techniques could useful to spot hidden patterns in data and reduce dimensionality in order to find outliers. One emerging approach involves Autoencoder, a particular type of Deep Neural Network.

1 Background

1.1 Autoencoder

An autoencoder, is a neural network used for unsupervised learning. This model uses back-propagation setting the target values to be equal to the inputs. The aim is to learn the identity function by reducing input dimensionality in a hidden space.

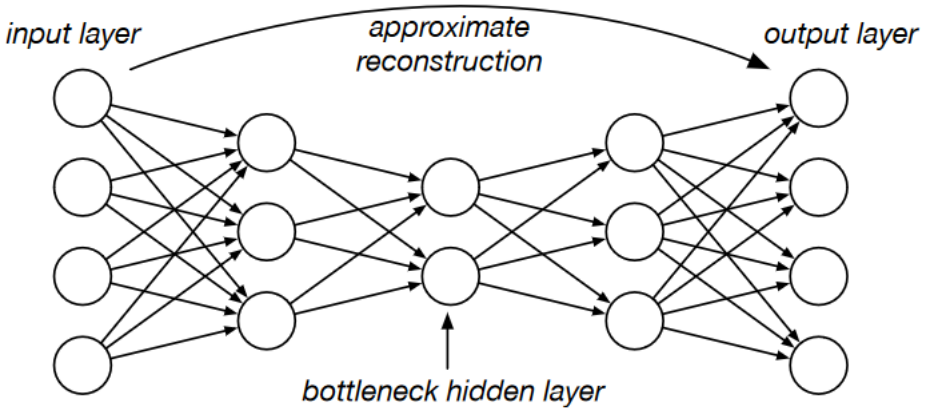


Figure 1.1: Autoencoder structure[6]

Architecturally the simplest form is a feedforward non-recurrent neural network. It is a particular form of a multilayer perceptron, where the output layer has the same number of nodes as the input layer and it consists of two parts: encoder and decoder. Mathematically, an autoencoder is defined as a tuple [7]:

$$(\mathbb{F}^n, \mathbb{G}^p, \mathfrak{A}, \mathfrak{B}, X, \Delta)$$

where:

1. F and G are sets
2. $n, p \in \mathbb{N}, 0 < p < n$
3. $\mathfrak{A}$ is a class of function from $\mathbb{F}^n$ to $\mathbb{G}^p$
4. $\mathfrak{B}$ is a class of function from $\mathbb{G}^p$ to $\mathbb{F}^n$
5. $X = \{x_1, ..., x_m\}$ is a set of m training vectors in $\mathbb{F}^n$
6. Δ is a dissimilarity, or distortion function, defined over $\mathbb{F}^n$

The autoencoder problem is to find $A \in \mathfrak{A}, B \in \mathfrak{B}$ such that :

$$\min E(A,B) = \min_{A,B} \sum_{t=1}^{m} E(x_t) = \min_{A,B} \sum_{t=1}^{m} \Delta(A \circ B(x_t), x_t)$$

When p < n the autoencoder tries to implement some form of lossy compression or feature extraction. The loss of information is unavoidable. Since it is minimized for particular homogeneous input data, when the autoencoder tries to reconstruct a different type of instance the error will be greater than the one obtained during the training.

2 Experimental results

2.1 Dataset

The dataset used in this study case is composed by 284'807 transactions made in two days of September 2013 by European cardholders found on Kaggle [4][10]. Due to confidentiality issues the original features' names are not provided and are the result of PCA. Only two features are provided with a description and refer to the time elapsed between each transaction and the fraud flag. Although frauds have been already marked, the dataset is highly unbalanced because they are only 492 out of 284'000. This kind of problem is difficult to approach with

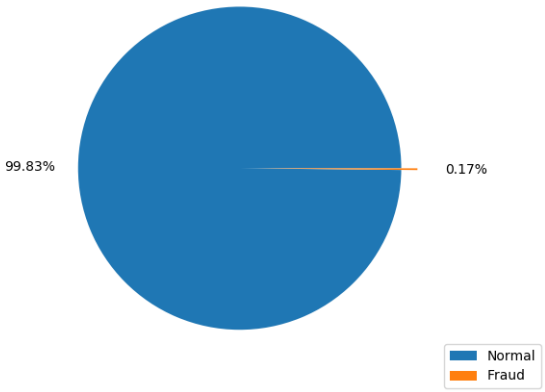


Figure 2.1: Dataset composition, 0.17% is clearly too low to create a non-biased classifier with traditional techniques

traditional Machine Learning classification techniques because performances get biased towards majority classes. Those algorithms are accuracy driven: they aim to minimize the overall error assuming that the datasets have balanced class distributions. In this case the most important class is the fraud class, but there are not enough fraud instances to understand fraud data underlying structure. Autoencoders could instead help to extract hidden patterns in non-fraudolent data. At the end of the learning it will be able to reconstruct non-fraudolent-like data with a low error reconstruction, while it should be high in reconstructing fraudolent data.

2.2 PREPROCESSING

First the whole .csv dataset is loaded into a Pandas dataframe, which allows to manipulate and making queries on data. Feature scaling can be an important preprocessing step for many machine learning algorithms. Amount column has a wide range of values so the neural network will not work well without a feature scaling. Standardization involves rescaling the features such that they have the properties of a standard normal distribution with a mean of zero and a standard deviation of one. This process is made with the scikitlearn class StandardScaler.

```
count     284807.000000  |       count    2.848070e+05
mean          88.349619  |       mean     3.202236e-16
std          250.120109  |       std      1.000002e+00
min            0.000000  |       min      -3.532294e-01
25%            5.600000  |       25%      -3.308401e-01
50%           22.000000  |       50%      -2.652715e-01
75%           77.165000  |       75%      -4.471707e-02
max        25691.160000  |       max       1.023622e+02
```

Listing 1: descriptive statistics that summarize the central tendency, dispersion and shape of Amount column distribution before and after the scaling

The next step is to split the dataset in two parts: *normal* and *fraud*, containing normal and fraudolent transaction according to the class column value. The first and the last columns (Time and Class) are not useful for the model that is be going to be built, so they are removed. Finally the training and the test sets are created. The normal transaction set is splitted, allocating the 80% for the training and the 20% for the test set. This test set is splitted again because its 90% will be used for computing the mean reconstruction error in the autoencoder training, while the remaining 10% will be used, combined to all fraudolent data, to evaluate the classifier.

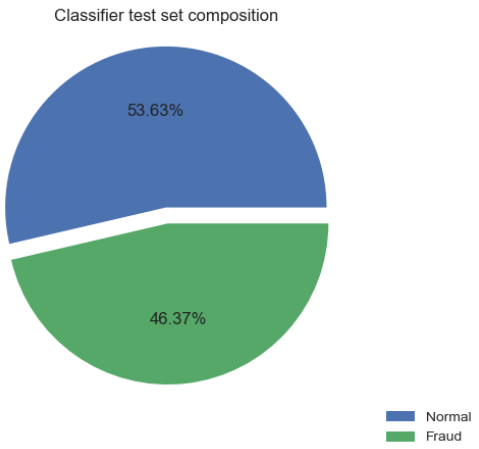


Figure 2.2: Test set structure

2.3 TOPOLOGIES OF NETWORKS USED

For the experiment two autoencoders with different number of layers have been built to understand and compare the performance of classification using different parameters. The number of neurons in each layer is splitted in halves going deeper in the network. The first model has three inner layers, which implies that the codification is represented by $\lfloor \frac{29}{2^2} \rfloor = 7$ neurons, the second model has 5 hidden layers, with a code made by $\lfloor \frac{29}{2^3} \rfloor = 3$ neurons. The

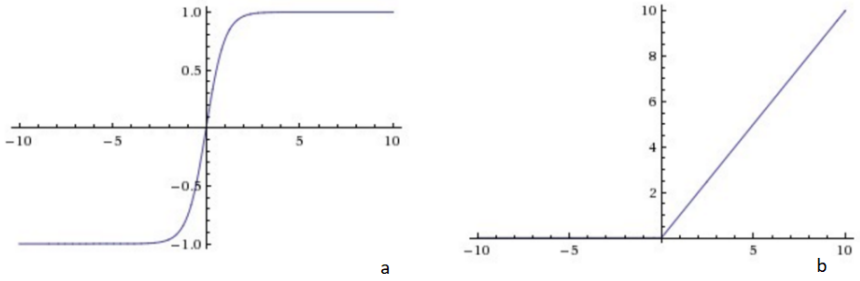
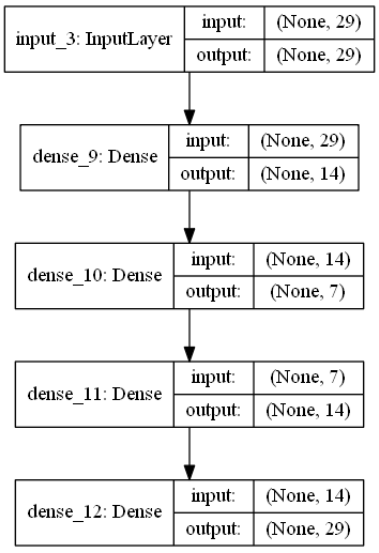
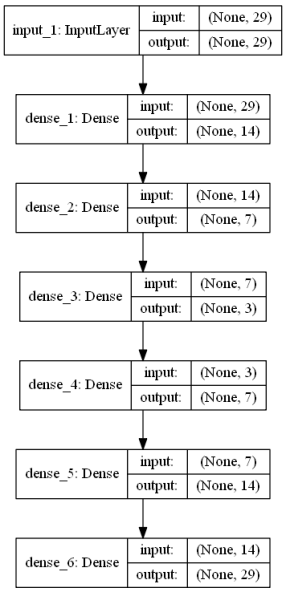


Figure 2.3: A comparison between the tanh (a) and the ReLU (b) activation functions.

ReLU and tanh activation functions are used alternately in the hidden layer in order to have the tanh as the activation function for the output layer. The tanh is bounded to [-1, 1], while ReLU, defined as $ReLU(x) = max(0, x)$, has a range of [0, inf) this means it could blow up the activation. The functions are both non linear, so that their combination will also learn a non-



(a) The first model, which involves 1'072 trainable parameters



(b) The second model, which involves 1,124 trainable parameters

linear function. ReLU involves simpler mathematical operations, so it is less computationally expensive and its activation is sparse, due to the random weight initialization. The loss function used for the neural network consists in the Mean Absolute Error, denoting with $\hat{x}$ the predicted reconstruction and with x the original input:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{x}_i - x_i|$$

The optimizer used in this case is Adam, a method for efficient stochastic optimization that only requires first-order gradients with little memory requirement. The method computes individual adaptive learning rates for different parameters: parameters that would ordinarily receive smaller or less frequent updates receive larger update with Adam [8]. This speeds learning in cases where the appropriate learning rates vary across parameters. In this particular case gradients could become very small at early layers because the tanh is used, so it makes sense to increase learning rates for the corresponding parameter.

2.4 Training and test

While training through epochs, the loss function converged rapidly to a very low value as shown in figure2.5. Predictably the first model's errors are very low. This happens because

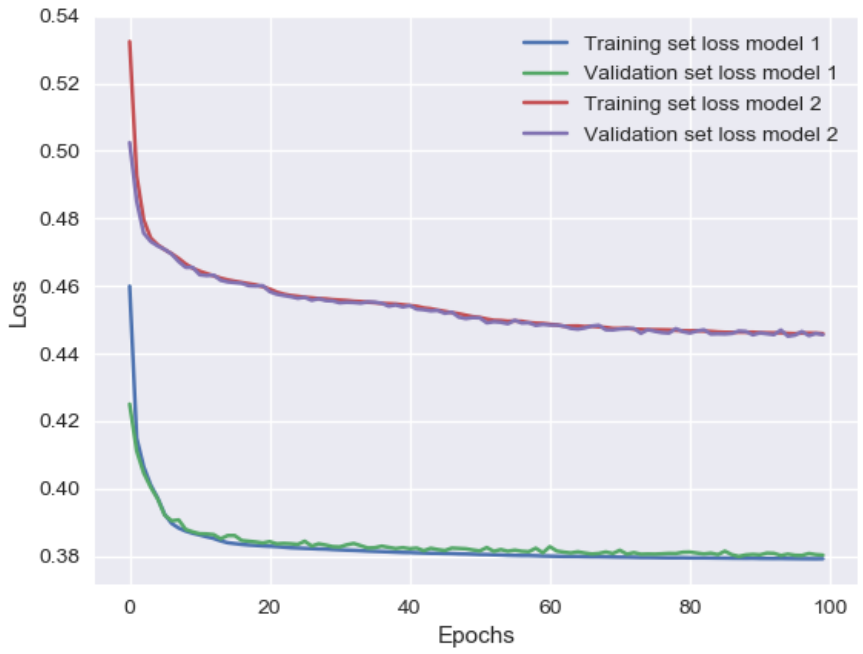


Figure 2.5: Data reconstruction errors during training

hidden layers, that act like a form of intermediate lossy compression, are less than the second

model. At the end of the training the autoencoders will be presumably able to reconstruct non-fraudolent transaction respectively with an error of $e_1 \approx 0.38380949677564702$, $e_2 \approx 0.45356815268955297$, obtained from the mean validation loss error. The next step is to find an opportune threshold value, beyond which an instance could be classified fraudolent or not, that maximize the recall and precision. To achieve this, several tolerance values to subtract

(a) First model: the best tolerance is between 0.1 and 0.12

(b) Second model: the best tolerance is between 0.13 and 0.15

Figure 2.6: Metrics comparison for different tolerance values between 0 and the mean validation loss for the different models

from the mean validation error are computed between 0 and the mean loss itself. For this type of problem is important to maximize recall, in order to find all frauds, and precision, in order to minimize a hypothetic further human effort. The best tolerance value could be, therefore, the intersection point between precision and recall. This is found by plotting those values in relation to tolerance as showed in figure 2.6. Now it is important to compare the different precision and recall values for each model respectively with tolerance 0.115 and 0.135 in order to find the best model.
According to the confusion matrix 2.3 for the specified tolerance values, precision an recall appear to be the same, but the second model, the one with a greater average reconstruction loss, gave a slightly better performance:

$$precision_1(0.115) = recall_1(0.115) = 0.8780487804878049$$

$$precision_2(0.135) = recall_2(0.135) = 0.8963414634146342$$

3 Conclusion

After preprocessing the dataset, built and trained with the same data two different models with different "compression rate" and compared the results, the model with the better performances turned out to be the one with more hidden layers and the larger reconstruction loss during the training.

However different architectures with different number of layers, activation functions, optimizers or loss function could be compared to this model. The only purpose of this project

Table 2.1: Model 1, tolerance = 0.115

		Prediction outcome		
		p	**n**	**total**
actual value	**p′**	432	60	492
	n′	60	509	569
	total	492	569	

Table 2.2: Model 2, tolerance = 0.135

		Prediction outcome		
		p	**n**	**total**
actual value	**p′**	441	51	492
	n′	51	518	569
	total	492	569	

Table 2.3: Confusion matrices

was to show that the simplest form of autoencoders could be used to approach this kind of problem involving outliers detection or binary classification with a very large dataset unbalance. The proposed models could be compared to a variational autoencoder [9]. A VAE uses a probabilistic encoder, and decoder, trained according to a reconstruction probability which can be used more objectively to classify instance as an outlier or not.

Reconstruction error is difficult to calculate if input variables are heterogeneous and there is no clear objective threshold in this case. Reconstruction probability does not require weighting the reconstruction error and is more objective because it is expressed in percentage.

REFERENCES

[1] G.K. Palshikar, The Hidden Truth, *Frauds and Their Control: A Critical Application for Business Intelligence, Intelligent Enterprise*, 2002

[2] R. Bolton, D. Hand *Statistical Fraud Detection: A Review (With Discussion)* Statistical Science 17(3): 235–255. 2002

[3] A. Dal Pozzolo, O. Caelen, Y. Le Borgne, S. Waterschoot, G. Bontempi. *Learned lessons in credit card fraud detection from a practitioner perspective*. Expert systems with applications 41: 10 4915–4928. 2014

[4] A. Dal Pozzolo, O. Caelen, Reid A. Johnson, G. Bontempi *Calibrating Probability with Undersampling for Unbalanced Classification* In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE 2015

[5] R. Bolton, D. Hand *Unsupervised profiling methods for fraud detection, Credit Scoring and Credit Control VII* 235–255. 2001

[6] Jinghui Chen, Saket Sathe, Charu Aggarwal, Deepak Turaga *Outlier Detection with Autoencoder Ensembles* 2017

[7] Pierre Baldi *Autoencoders, Unsupervised Learning, and Deep Architectures*

[8] Diederik P. Kingma, Jimmy Lei Ba *Adam, a Method for Stochastic Optimization*

[9] Jinwon An, Sungzoon Cho *Variational Autoencoder based Anomaly Detection using Reconstruction Probability*

[10] *https://www.kaggle.com/dalpozz/creditcardfraud*