Bachelor thesis
Computer Science

Radboud University

Autoencoding Credit Card Fraud

Author:
Tom Sweers
S4584325

First supervisor/assessor:
prof. dr. Tom Heskes
t.heskes@science.ru.nl

Second assessor:
Jesse Krijthe
jkrijthe@gmail.com

June 26, 2018

Abstract

We propose a credit card fraud detection method using autoencoder and variational autoencoder based anomaly detection. Autoencoders are neural networks that learn to encode data efficiently, and a variational autoencoder is a variant of autoencoder that uses a probabilistic graph as a basis. We used the reconstruction error as an anomaly score for the autoencoder and a reconstruction probability, the chance of generating the original data, for the variational autoencoder.

We used these principles to detect credit card fraud. Experimental results show that both perform decent in detecting fraud on our used data set and that overall regular autoencoders perform better than variational autoencoders.

Contents

Chapter 1

Introduction

Large purchases are often made by using credit cards. A risk for both companies and customers is a person using the credit card of someone else to pay. This is credit card fraud. Because fraud does relatively not happen often it is a difficult task recognizing fraudulent transactions.

In this paper we will propose a possible method to recognise credit card fraud by doing anomaly detection using autoencoders and variational autoencoders, and see which one performs better. An anomaly is a data point that is sufficiently different from the other data points [1]. Because fraud happens rarely it can be considered an anomaly and thus anomaly detection can be used to detect fraud. Autoencoders are neural networks that learn efficient encoding of data in such a way that the decoding is similar to the original data [2]. When an autoencoder is trained on data containing no anomalies the reconstruction of an anomaly will be worse than the reconstruction of a normal point, hence we can use this as an anomaly score [3].

Variational autoencoders are probabilistic variants of an autoencoder. We can use the reconstruction probability, the chance of generating the original data from the encoding, as the anomaly score [4].

First we will explain the concepts of autoencoders and variational autoencoders. Then we will give a short overview of anomaly detection and explain the combination of anomaly detection with the autoencoders and how this can be used for fraud detection. We will continue by explaining the method of the experiment. Then the results of the experiments are given and we will end with discussing related work and the results.

Chapter 2

Preliminaries

2.1 Anomaly Detection

Anomaly or outlier detection is the detection of points in a data set that are different from all other "normal" points [1]. It is often used in tasks such as intrusion detection (in cyber-security), fault detection in various types of systems and military surveillance. There are six types of techniques used to detect anomalies: Classification based techniques, clustering based techniques, statistical techniques, information theoretic techniques and spectral techniques [1].

Classification based methods use a labeled training set to learn and can then use this to classify all entries in a test set. One or more of the resulting classes can be considered anomalies [1].

Clustering uses unsupervised learning to group data points that are similar into a cluster. One can use this to detect anomalies by, for example, classifying points that are not in a cluster, or an entire cluster as anomalies [1].

Statistical methods use probabilistic distributions. When data is modeled by such a distribution one can classify a data point as an anomaly when it has a low (below a threshold) probability of being generated by the model [1].

Information theoretic techniques use the information content of a data set and apply information theoretic measures, like Kolomogorov Complexity and entropy. One can then find irregularities in the information content and find anomalies [1].

Spectral methods use dimension reduction of data. The principle is that anomalies look different from normal data points in this reduced form and can then be detected [1].

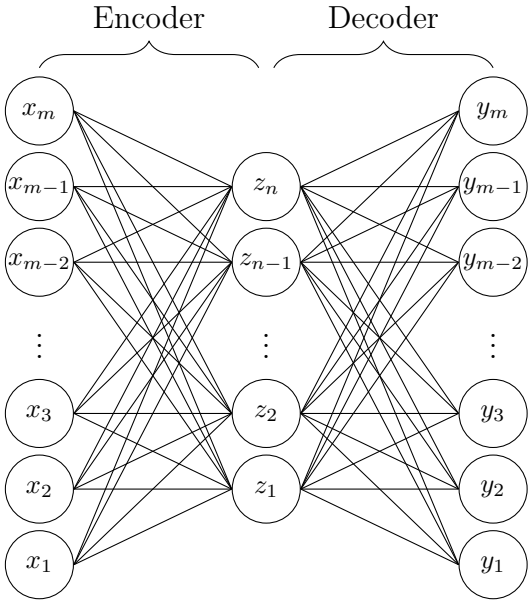

Figure 2.1: An autoencoder

2.2 Autoencoders

Autoencoders are neural networks that learn how to find an encoding of data so that the input and the reconstruction of the encoding are close to each other [5]. In its simplest form, the autoencoder's neural network structure consists of three layers like depicted in Figure 2.1. The first layer is the input layer and consists of m nodes, the same amount of nodes as the dimension of the data, and the middle hidden layer has n nodes, where n is the dimension of the encoding. The third layer is the output layer and, just as the input layer, it has m nodes. Typically autoencoders are used for dimensionality reduction, so the resulting encoding usually has a smaller dimension than the input data [2].

An autoencoder can be divided into two parts, an encoder and a decoder. The transition between the first and second layer represents the encoder and the transition between the second layer and the third layer represents the decoder. The equation

$$z = \sigma(W_{enc}x + b_{enc})$$

represents the encoder in this autoencoder [5]. The W is a n by n matrix consisting of the weights of the layer in the neural network. The b is a n-sized vector containing the bias of the layer. x is the input vector of length n. The σ stands for a nonlinear transformation function like the sigmoid function. In the same way we can represent the decoding

$$y = \sigma(W_{dec}z + b_{dec})$$

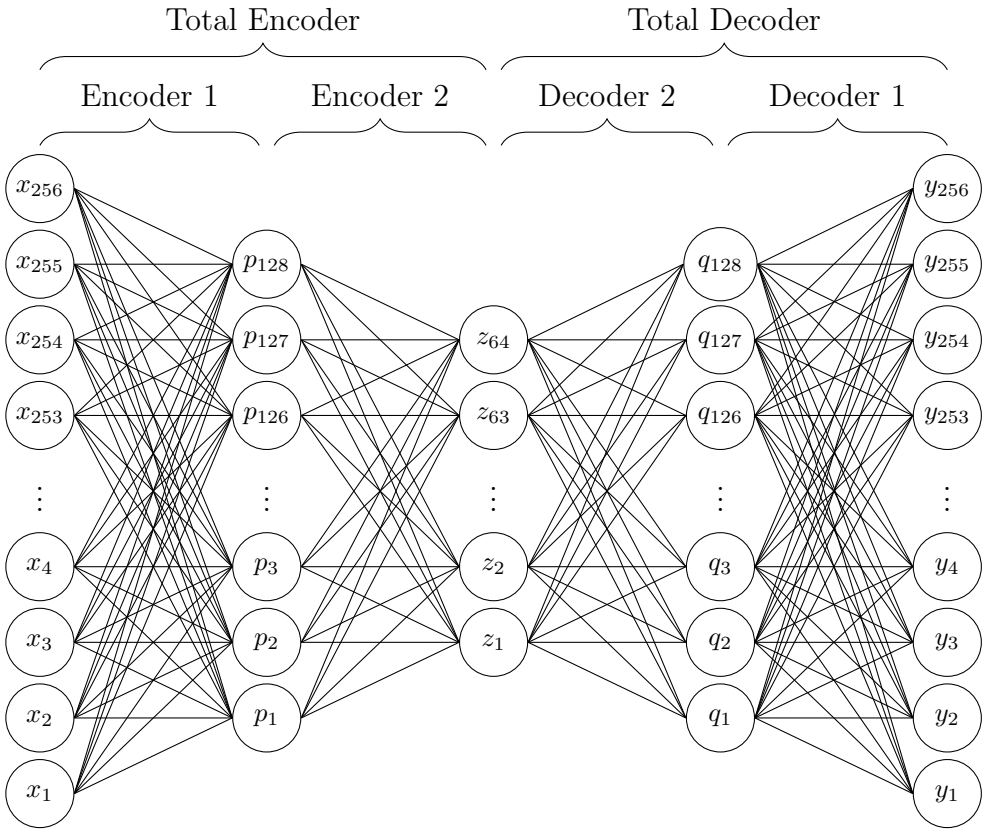

Figure 2.2: A stacked autoencoder

An autoencoder can also have a more complex structure by having more hidden layers, for example by stacking encoders and decoders [6]. For example consider the autoencoder shown in Figure 2.2, here the first encoder reduces the dimension from 256 to 128 and the second encoder from 128 to 64. Vice versa the first decoder enlarges the dimension back to 128 and the second one to 256. The equations of the entire neural network then looks like this

$$\begin{aligned} e_1 &= \sigma(W_{enc1}x + b_{enc1}) \\ e_2 &= \sigma(W_{enc2}e_1 + b_{enc2}) \\ d_1 &= \sigma(W_{dec1}e_2 + b_{dec1}) \\ y &= \sigma(W_{dec2}d_2 + b_{dec2}) \end{aligned}$$

An autoencoder learns, unsupervised, to make an encoding and decoding while minimizing the reconstruction error by a backpropagation algorithm. The reconstruction error is a metric measuring the similarity between input

and output of the autoencoder. During training the autoencoder is given unlabeled training data and it tries to minimize the reconstruction error over all training data. A commonly used reconstruction error is the mean squared error.

2.3 Variational Autoencoders

A variational autoencoder (VAE) is another version of a normal autoencoder. The difference is that a VAE is a discrete probabilistic graphical model (DPGM), while a regular autoencoder is a non probabilistic graphical model [4]. A neural network is used to approximate the posterior of the DPGM which results in a structure that resembles an autoencoder as shown in Figure 2.1 The approximate posterior of the DPGM is noted as $q_\phi(z|x)$, where x is the input data and z is a latent variable representing the encoded data, and $p_\theta(z)$ is the prior distribution of z [7]. The model of $p_\theta(x|z)$ functions as the decoder and the model of $q_\phi(z|x)$ functions as the encoder. These are trained by the neural network, whose goal is finding an optimal variational lower bound of the marginal likelihood of the input data [4].

Hence the neural network part of the VAE does the same thing as in a regular autoencoder, it learns, unsupervised, to minimize a reconstruction error. Only the reconstruction error that the autoencoders tries to minimize is different for the VAE. The reconstruction error of a VAE is calculated using the marginal likelihood of the data, which is the sum of the likelihood of all individual data points. During training the parameters of the VAE are tuned to maximize

$$Q = -D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) + E_{q_\phi(z|x^{(i)})}[\log p_\theta(x|z)]$$

where the first term is the Kullback-Leibler divergence of the approximate posterior and prior and the second term relates to the reconstruction of the data [7].

This is not the only difference between a regular autoencoder and a VAE. The parameters of the VAE that are being learned by the neural network are not parameters of the value, like in the case of a regular autoencoder, but of the distribution [4]. This means that the encoder and decoder are a so-called probabilistic encoder and decoder. Hence the encoder $f(x, \phi)$ does not output encoded data or the latent variable z, but instead it outputs parameters for the approximate posterior $q_\phi(z|x)$. To obtain z we have to sample from a normal distribution and reparameterize the sample to make it follow the distribution of the posterior [4]. This also holds for decoding using the probabilistic decoder.

Chapter 3

Research

3.1 Method

To recognise fraud in the credit card data set we are going to use a method that can be categorised as a spectral method: Anomaly detection by autoencoders and VAE's. Autoencoders reduce the dimension of the data and, as we will see, we can use the reconstruction error to measure the difference between anomalies and normal points in the reduced form. Hence we can use the reconstruction error as an anomaly score.

3.1.1 Anomaly Detection Using Autoencoders

The principle of anomaly detection by using autoencoders is as follows: First train the autoencoder on a training set that only contains the non-anomaly points. Because the autoencoder is now fitted on the normal data set, and the autoencoder has not yet encountered any anomalies, the reconstruction should perform better on normal data than on anomalies. Hence we can determine whether data points are anomalies or not by calculating the reconstruction error used to train the autoencoder. Generally the reconstruction errors of anomalies are higher than the reconstruction error of the normal data. Hence we can determine an upper bound α for the reconstruction error. Is the reconstruction error of point x^i higher than α then x^i is an anomaly, otherwise it is not [3].

3.1.2 Anomaly Detection Using Variational Autoencoders

Anomaly detection using VAE is based on the same principle as discussed before. The difference is that we don't use the reconstruction error but a reconstruction probability [4]. This is the probability that the data point can be generated from a latent variable drawn from the approximate posterior distribution. We calculate the reconstruction probability for data point x^i by first encoding x^i using the probabilistic encoder from the VAE, which

results in the distribution of the posterior. Then we draw L samples from a normal distribution and reparameterize it with the mean and standard deviation of the distribution of the posterior [4]. The probabilistic decoder is then used on the samples which results in a distribution of the prior. By using the mean and standard deviation of the resulting distribution for every sample we can calculate the reconstruction probability through

$$\frac{1}{L}\sum_{l=1}^{L} p_\theta(x^{(i)}|\mu_{x^{(i,l)}}, \sigma_{x^{(i,l)}})$$

We can again determine a boundary α to evaluate whether x^i is an anomaly or not. When the reconstruction probability of a data point is above α then the chance of reconstructing it from the encoder is high, so it is not an anomaly. When the reconstruction probability is lower or equal it is an anomaly.

3.1.3 The Experiment

The Data

The data was collected from the data mining contest website Kaggle [8]. It consists of two days of credit card transactions from Europeans done in September 2013. There are a total of 284,807 transactions of which 492 are considered fraud. This is, as said earlier, highly unbalanced.

All transactions are labeled as either fraudulent or not. Each transaction consists of 30 features. The first 28 features where obtained by a PCA transformation. What the original features were and any other information about it could not be given because of the sensitivity and confidentiality issues with the data. The other two features are "time" and "amount" corresponding respectively with the point of time the transaction occurred and the amount of money that was paid.

Before training all features were normalized and the data set was split in a training and a test set. We sampled 80% of the normal transactions of the data set to serve as the training set. The other 20% and the fraudulent transactions are used as the test set.

The autoencoders

We built four different autoencoders models and four different variational autoencoder models. The first two consist of neural networks having only one hidden layer. The other two are stacked autoencoders: They consist of 5 hidden layers with the middle one being the smallest. The difference between these two are the amount of nodes in these layers.

In case of the one hidden layer network the amount of nodes in this one layer is either 2 or 10. In the case of the stacked autoencoders the sequence

of the amount of nodes in the hidden layers in the first network is: 25-20-10. The other network has this sequence: 20-10-2.

To build and train these autoencoders we used Tensorflow in Python. We used 0.01 as learning rate and trained every neural network with 100 epochs. Every model is trained and tested 10 times to ensure that the results are valid. The scripts for the autoencoder are based on [9] and the scripts for the VAE's are based on [10].

Performance

To measure the performance of the autoencoders and VAE's we used recall-at-k and precision-at-k:

$$\text{recall-at-k} = \frac{|F_{True} \cap PF_k|}{|F_{True}|}$$
$$\text{precision-at-k} = \frac{|F_{True} \cap PF_k|}{|PF_k|}$$

where F_{True} is the set containing all true fraudulent transactions and PF_k is the set containing the top k predicted fraud cases, sorted in descending order on the reconstruction error or reconstruction probability.

We did not use measures like accuracy because these give a wrong view of the situation due to the imbalance of the data.

3.2 Results

3.2.1 Autoencoder

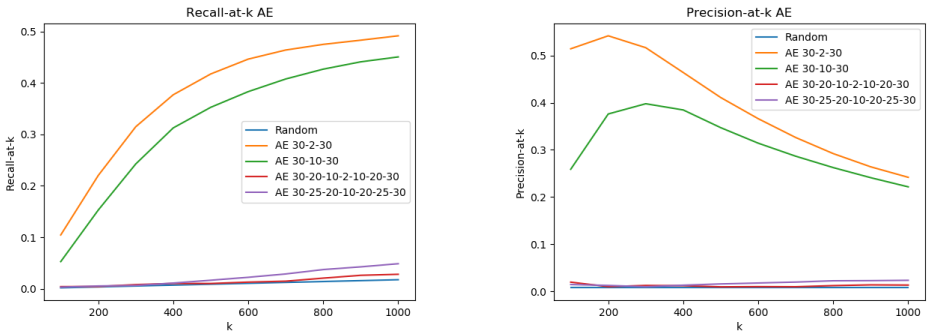

Figure 3.1: Left: A graph showing recall-at-k for the autoencoders. Right: A graph showing precision-at-k for the autoencoders.

Autoencoder	**Recall at 10000**	**Recall at 20000**	**Recall at 50000**
Random	0.174	0.349	0.872
AE (30-2-30)	0.565	0.621	0.856
AE (30-10-30)	0.556	0.613	0.856
AE (30-20-10-2-10-20-30)	0.545	0.632	0.898
AE (30-25-20-10-20-25-30)	0.617	0.699	0.938

Table 3.1: Recall-at-k scores for the regular autoencoders

Autoencoder	**Precision at 10000**	**Precision at 20000**	**Precision at 50000**
Random	0.008	0.008	0.008
AE (30-2-30)	0.028	0.015	0.008
AE (30-10-30)	0.027	0.016	0.008
AE (30-20-10-2-10-20-30)	0.027	0.016	0.009
AE (30-25-20-10-20-25-30)	0.030	0.017	0.009

Table 3.2: Precision-at-k scores for the regular autoencoders

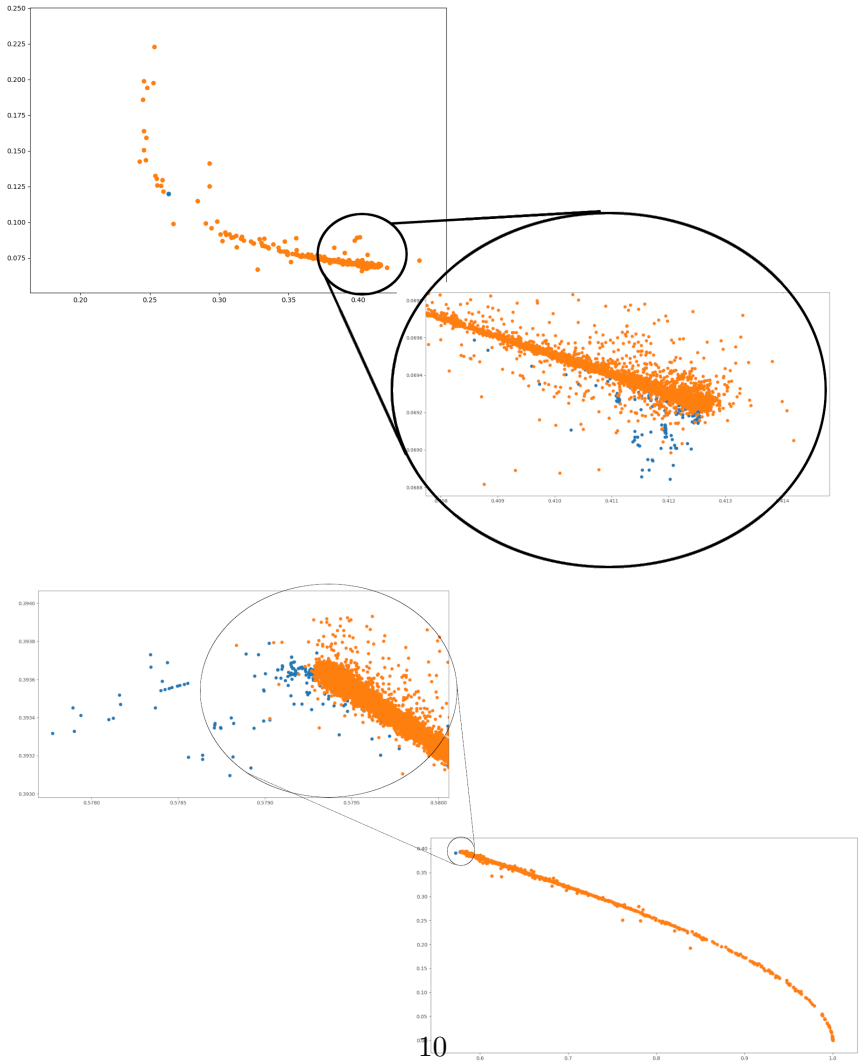

Figure 3.2: Scatterplot of the encoding of the test data by the 30-2-30 autoencoder (above) and the 30-20-10-2-10-20-30 autoencoder (below). The orange dots are normal transactions and the blue points are fraud.

Figure 3.1 shows the recall-at-k and precision-at-k for all autoencoders on the credit card data set. It shows that for k lower than 1000 the simpler autoencoders, the ones with one hidden layer, perform better than the stacked autoencoders. In the case of the simple autoencoder the one that reduces the dimension to 2 detects fraud better than the autoencoder that reduces the dimension to 10. That difference is not notable for the stacked autoencoders. The stacked autoencoders also are barely better than a random detector, while the simple autoencoders perform significantly better than random.

Table 3.1 shows the recall-at-k of all four autoencoder models on the credit card data set for higher values of k. Although less relevant than the lower values of k it still shows that the stacked autoencoders outperform the simpler autoencoders with only one hidden layer when k is higher than 20000, and the stacked autoencoder that reduces the dimension to 10 is already better at $k = 10000$. The same can be concluded from the precision-at-k in Table 3.2.

The differences in the dimension that the autoencoder reduces to are also notable at the higher values, although the difference in the simpler autoencoder becomes less significant. The difference grows actually more significant for the stacked autoencoders. But for the stacked autoencoders the difference is the other way around; the autoencoder that reduces the dimension to 10 performs better than the other.

Another notable difference is the average reconstruction error on the test set. The reconstruction error is significantly lower with the stacked autoencoders than with the regular autoencoders even with the same amount of training time. It is worth mentioning that during training this difference is immediately noticeable, in the first epoch the reconstruction error on the simple autoencoder starts around 0.30 and the reconstruction error on the stacked autoencoder starts around 0.030.

Figure 3.2 shows scatterplots of the encoding of the autoencoders that reduce the dimension to 2. The scatterplot of the more complex autoencoder has more distinguishable fraudulent data points, which resembles the overall better results of the more complex autoencoder for higher values of k.

3.2.2 Variational Autoencoder

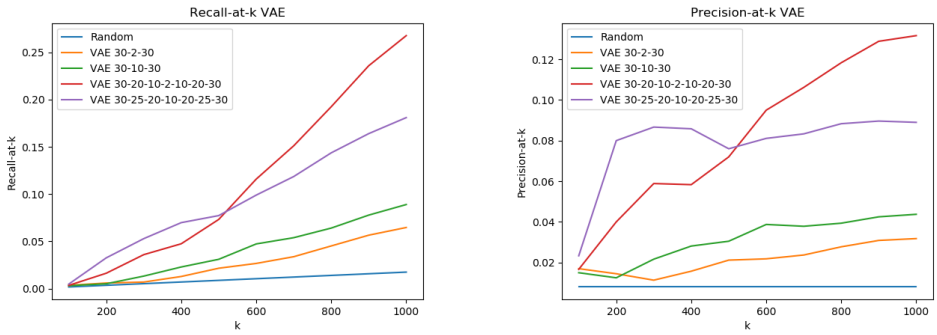

Figure 3.3: Left: A graph showing recall-at-k for the VAE's. Right: A graph showing precision-at-k for the VAE's.

VAE	**Recall at 10000**	**Recall at 20000**	**Recall at 50000**
Random	0.174	0.349	0.872
VAE (30-2-30)	0.529	0.688	0.883
VAE (30-10-30)	0.498	0.652	0.901
VAE (30-20-10-2-10-20-30)	0.736	0.820	0.913
VAE (30-25-20-10-20-25-30)	0.536	0.600	0.800

Table 3.3: Recall-at-k scores for the variational autoencoders

VAE	**Precision at 10000**	**Precision at 20000**	**Precision at 50000**
Random	0.008	0.008	0.008
VAE (30-2-30)	0.026	0.017	0.009
VAE (30-10-30)	0.025	0.016	0.009
VAE (30-20-10-2-10-20-30)	0.036	0.020	0.009
VAE (30-25-20-10-20-25-30)	0.026	0.015	0.008

Table 3.4: Precision-at-k scores for the variational autoencoders

In Figure 3.3 we can see the recall-at-k and precision-at-k scores for the VAE's. In contrast with the regular autoencoders we can see that stacked VAE's score significantly better than one-layered VAE's when $k \leq 1000$. Also in contrast with the regular autoencoder is the difference in score when varying the amount of hidden nodes. For the VAE's with one hidden layer, the VAE reducing the dimension to 10 performs better, while it was the other way around for the regular autoencoders. For the stacked VAE's the same is true when $k \leq 500$, when k is larger the VAE reducing the dimension to 2 is better. All VAE's perform better than a random detecting method.

In Table 3.3 and Table 3.4 we can see the scores of the VAE's for higher values of k. We can see that the stacked VAE's also perform than the simple VAE's when k is higher, although there is a clear difference in score between the stacked VAE that reduces the dimension to 2 and the stacked VAE that reduces the dimension to 10. For both the stacked and the one-layered VAE's the model that reduces the dimension to 2 is the better one.

With k under 10000 the best regular autoencoders (the autoencoders with one hidden layer) have better scores than the best VAE's (the stacked VAE's). All four VAE's perform better than the stacked autoencoders however. The stacked VAE's are as good as the regular autoencoders when k gets higher than 10000. The VAE with sequence 30-20-10-2-10-20-30 actually scores higher than the regular autoencoders. This difference is less noticeable when k becomes higher than 20000.

Chapter 4

Related Work

In this paper we proposed a method to detect credit card fraud. In this section we will discuss other researched methods.

Credit card fraud detection has previously been done by methods like

- Neural and Bayesian Networks
- Logistic Regression
- Decision Tree
- Hidden Markov Model
- K-Nearest Neighbour Algorithm

Roy et al [11] used four different methods to detect credit card fraud. They analysed and tested a Feed Forward Multilayer Perceptron network, Recurrent Neural Networks, Long Short-term Memory and Gated Recurrent Unit. They found that the Gated Recurrent Unit performed the best. For each of the methods they concluded that larger networks perform better.

Iyer et al [12] used a Hidden Markov Model (HMM) to detect credit card fraud. In this paper the HMM is trained on a training set containing only legitimate transactions. The authors conclude that method performs better than others, although hybrid models have to be used to avoid high false positives.

Awoyemi et al [13] tested three classifier based methods. From naive Bayes, k-nearest neighbour and logistic regression they found that the second performed best. They also concluded that training with a 34:66 (fraud against non-fraud) distribution improved results.

Malini and Pushpa [14] analysed k-nearest neighbours and anomaly detection methods to detect credit card fraud. They concluded that both methods work well, although k-nearest neighbours are generally more accurate and efficient.

Chen et al [3] used autoencoder-based anomaly detection to detect network cyber attacks. They ran an experiment on NSL-KDD data set, where they tested PCA, Autoencoder and convolutional autoencoder based anomaly detection. They concluded that convolutional autoencoders performs the best, even more they concluded that it outperformed other detection methods.

These papers are relevant because they researched credit card fraud detection, but they all used different data sets as each other and as us. On the website Kaggle, where the data set originates, multiple solutions have been proposed.

Kaggle user Neil Schneider [15] compared GBM, xgboost and lightGDM in detecting fraud in the credit card data set. GDM performed worst, while xgboost and lightGDM are close to each other with a small advantage for lightGDM. Besides the higher accuracy he also preferred lightGDM because of faster training time and low memory usage.

Another user named Gabriel Preda [16] experimented with RandomForest on the data set. He concluded that the performance, considering that he did little tweaking, was relatively good.

For more information on the anomaly detection using VAE's that we implemented [4] can be assessed. This paper explains the algorithm in great detail and it shows by experiment that the VAE's outperform the regular autoencoders on anomaly detection.

Chapter 5

Conclusions

We have applied anomaly detection by using autoencoders and VAE's to detect credit card fraud. In our experiment we have found that the simple regular autoencoders outperform the VAE's when detecting credit card fraud, although the stacked VAE's scored better than the stacked autoencoders and simple VAE's.

So when the autoencoders and VAE's are constructed the right way, they detect credit card fraud, in the data set we used, quite well. Because our analysis is only done on one data set it is not certain that our conclusions are also applicable on other credit card fraud sets.

So to further research this topic one could test our method on other credit card data sets and see if our conclusions also hold on these data sets. More research can also be put into detecting credit card fraud using autoencoders and VAE's with even more hidden layers.

All things considered autoencoders and VAE's perform decent in detecting credit card fraud.

Bibliography

[1] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[2] Yasi Wang, Hongxun Yao, and Sicheng Zhao. Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232–242, 2016.

[3] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau. Autoencoder-based network anomaly detection. In *2018 Wireless Telecommunications Symposium (WTS)*, pages 1–5, April 2018.

[4] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *SNU Data Mining Center, Tech. Rep.*, 2015.

[5] J. Zheng and L. Peng. An autoencoder-based image reconstruction for electrical capacitance tomography. *IEEE Sensors Journal*, 18(13):5464–5474, July 2018.

[6] M. M. Yang, A. Nayeem, and L. L. Shen. Plant classification based on stacked autoencoder. In *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pages 1082–1086, Dec 2017.

[7] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[8] A. D. Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi. Calibrating probability with undersampling for unbalanced classification. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 159–166, Dec 2015.

[9] Autoencoder tensorflow. `https://github.com/aymericdamien/TensorFlow-Examples/blob/master/examples/3_NeuralNetworks/autoencoder.py`. Accessed: 18-05-2018.

[10] Tensorflow mnist vae. `https://github.com/hwalsuklee/tensorflow-mnist-VAE`. Accessed: 12-06-2018.

[11] A. Roy, J. Sun, R. Mahoney, L. Alonzi, S. Adams, and P. Beling. Deep learning detecting fraud in credit card transactions. In *2018 Systems and Information Engineering Design Symposium (SIEDS)*, pages 129–134, April 2018.

[12] D. Iyer, A. Mohanpurkar, S. Janardhan, D. Rathod, and A. Sardeshmukh. Credit card fraud detection using hidden markov model. In *2011 World Congress on Information and Communication Technologies*, pages 1062–1066, Dec 2011.

[13] J. O. Awoyemi, A. O. Adetunmbi, and S. A. Oluwadare. Credit card fraud detection using machine learning techniques: A comparative analysis. In *2017 International Conference on Computing Networking and Informatics (ICCNI)*, pages 1–9, Oct 2017.

[14] N Malini and M Pushpa. Analysis on credit card fraud identification techniques based on knn and outlier detection. In *Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), 2017 Third International Conference on*, pages 255–258. IEEE, 2017.

[15] Gbm vs xgboost vs lightgbm. `https://www.kaggle.com/nschneider/gbm-vs-xgboost-vs-lightgbm`. Accessed: 25-06-2018.

[16] Credit card fraud detection with rf. `https://www.kaggle.com/gpreda/credit-card-fraud-detection-with-rf-auc-0-93`. Accessed: 25-06-2018.