Credit Card Fraud Detection using Autoencoder and Restricted Boltzmann Machine

Arun Sharma & Aman Malkar

Abstract –**The goal of this project is to develop an unsupervised machine learning algorithm that explores a data set of credit card transactions and is able to tell if a particular transaction is fraud or genuine. Since the model is unsupervised, we will train our model with both positive and negative data (i.e. fraudulent and non-fraudulent) without providing the labels. Since the dataset has more normal transactions than fraudulent ones, we expect the model to learn and memorize the patterns of positive ones and able to give a score for any transaction as being an outlier.**

I. Introduction to Fraud Detection

PwC economic crime survey of 2016 suggests that more than one in three (36%) of organizations experienced some form of economic crime. Traditional methods of data analysis have long been used to detect fraud. They often require complex, time-consuming investigations that deal with different domains of knowledge like finance, economics, business practices and law. Fraudulent practices often consist of many instances involving repeated transgressions using the same method. Most IT entities today that are in use are transactional. Which means that data transactions are processed in the system the all the details are stored in the systems database. The relationships and patterns in these transactions can be analyzed and used for various purposes. This gives us two options to analyze the transactions, either by using a supervised learning algorithm or an unsupervised learning algorithm. While Supervised learning algorithms use samples from data and consist of labels such as 'fraudulent' or 'non-fraudulent' classes as a base to create a model that would be able to detect new transactions as fraudulent or not, it's not available to have a dataset that would have a classified data with labels. Hence, we decided to use unsupervised learning to train a model that would not make use of labelled records and feed the model with transactions and tune it to distinguish between the transactions as being genuine or fraudulent.

II. Why not Supervised Learning?

While supervised learning algorithms are used in fraud detection, they rely heavily on labelled data i.e. one required to be confident about the true classes of original data to build the model that will efficiently classify new transactions properly. This adds an element of uncertainty when legitimate transactions are reported as fraud and fraudulent observations are not reported as such. Moreover, these methods also suffer from problem of unbalanced class sizes; in fraud detection problems, the legitimate transactions generally far outnumber the fraudulent ones and this imbalance can cause misspecification of models. In our own dataset, there are 284,807 transactions over a 48-hour period out of which only 0.17% are fraud and the rest are genuine. Hence, it would be wise not to use supervised learning for this dataset.

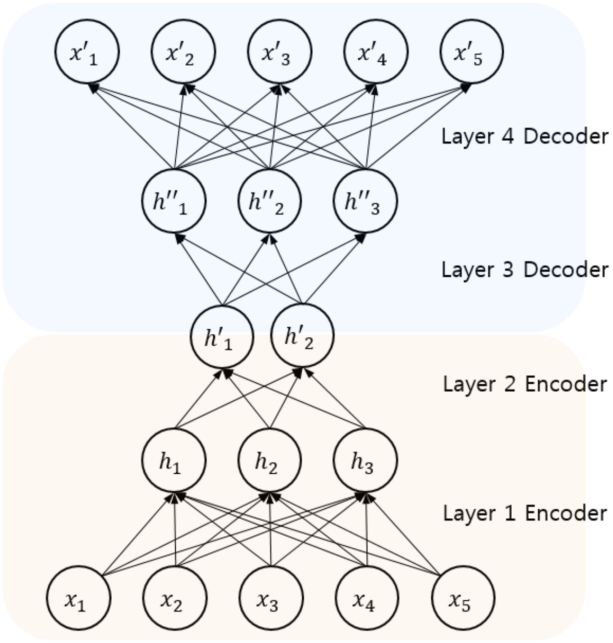
III. Approach – Unsupervised Learning

We have used two unsupervised learning models, namely, Autoencoder and Restricted Boltzmann Machine in TensorFlow to create models.

A. *Autoencoder*

Autoencoder is one type of neural network that approximates the function $f(x) = x$. Basically, given an input x, the network will learn to output $f(x)$ that is as close as x. The error between output and x is commonly measured using root mean squared error (RMSE) which is the loss function that we will try to minimize in our network model.

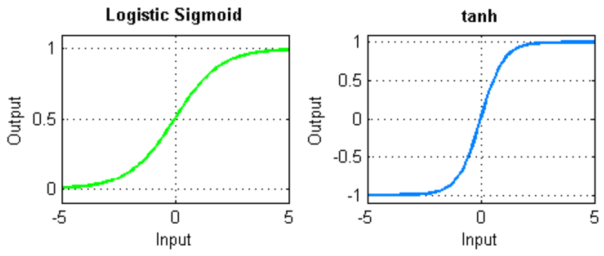
The Autoencoder follows a typical feed-forward neural network architecture except that the output layer has exactly same number of neurons as the input layer. And it uses the input data itself as its target. Therefore, it works in a way of unsupervised learning – learn without predicting an actual label.
An Autoencoder is represented below which follows the properties we described above and consists of encoder and decoder layers as described.



The lower part of the network is usually called an 'encoder' – whose job is to embed the input data into a lower dimensional array. The upper part of the network, or 'decoder', will try to decode the embedding array into the original one. We can have either one hidden layer or multiple layers depending on the complexity of our features.

We rely on the Autoencoder to 'learn' and 'memorize' the common patterns shared by majority of the training data. And during reconstruction, the RMSE will be high for data who do not conform for those patterns. And these are the 'anomalies' that we target to detect. And these anomalies are nothing but the 'fraudulent' transactions that we are after.

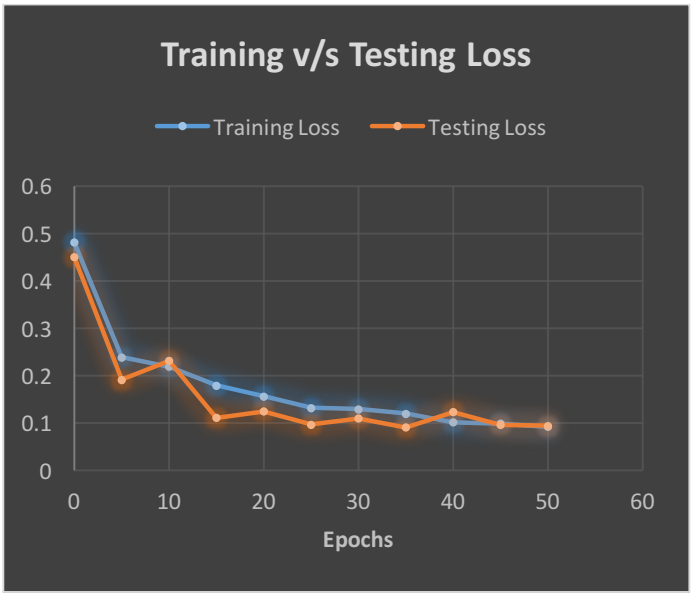
Since the data is unbalanced, we have used the first 75% as training and validation data while the latter 25% as testing data. For data standardization we have used z-score normalization because we are going to use tanh as our activation function.

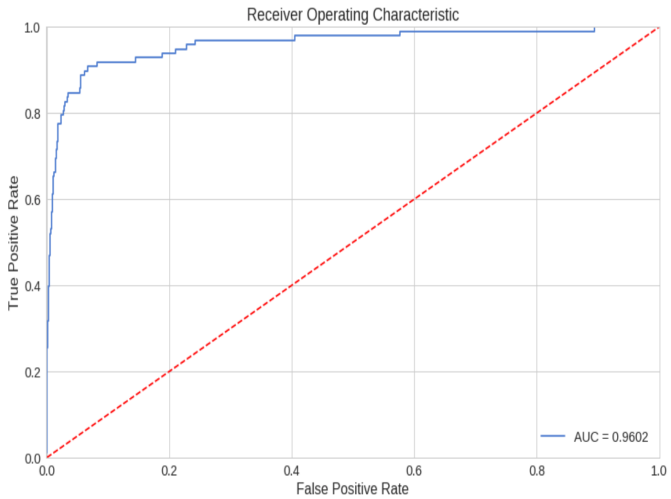


Sigmoid will squash the values in the range (0,1) whereas tanh squashes them into (-1,1) and in our validation set testing we found that tanh outperforms sigmoid. Hence, the decision.

The objective cost function measures the total RMSE of our predicted and input arrays in one batch which is a scalar and we run the optimizer every time we want to do a batch update. However, we also use a variable to return RMSEs for each input data in a batch which is a vector of length that equals number of rows in the data. Every time we sample data (mini batch) of size 256 from training set, we feed it into the model as input and run the optimizer to update the parameters.

We are using same data for training and validation, which does seem counter-intuitive at first but since the model never 'sees' the labels during training, it wont lead to overfitting.



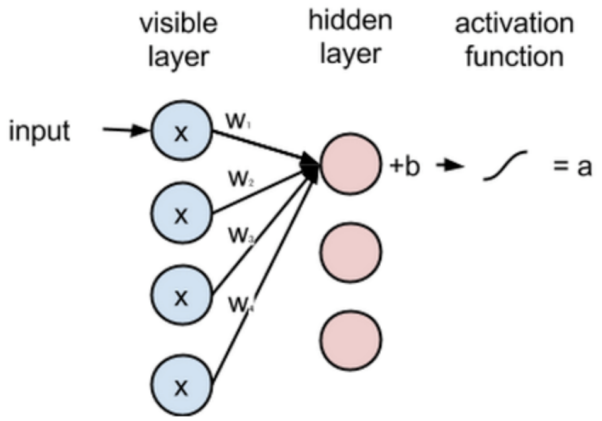


Since our evaluation metric is the AUC score on test data set, we can say that our model is good considering the 0.962 AUC score.

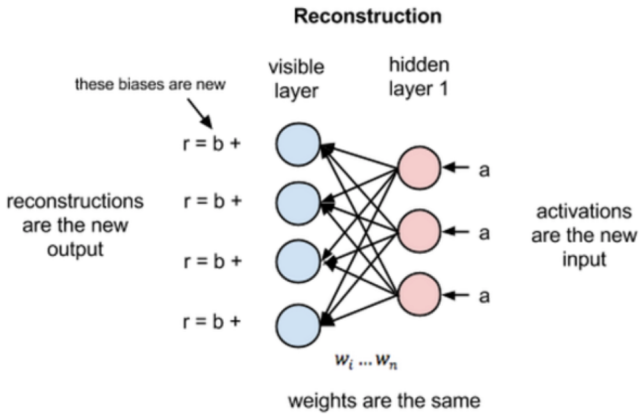
B. *Restricted Boltzmann Machine*

Restricted Boltzmann Machines were of the earliest models used in the world of deep learning and their applications range from dimensionality reduction, classification, collaborative filtering, feature learning to anomaly detection. We will use the same data set here as well.

Basically, a RBM network consists of two layers – the Visible Layer and the Hidden Layer. There are symmetric connections between any pair of nodes from Visible and Hidden Layers and no connections within each layer. For majority cases, both hidden and visible layers are binary-valued.



When signals propagate from visible to hidden layer, the input is multiplied with W matrix, added with bias vector b of hidden layer and finally squashed to be within 0 and 1.



During backward pass(reconstruction), the hidden layer activation becomes the input which is multiplied by the same matrix W along with the added bias.

To detect fraud with RBM, we will use the Free Energy derived from the energy function.

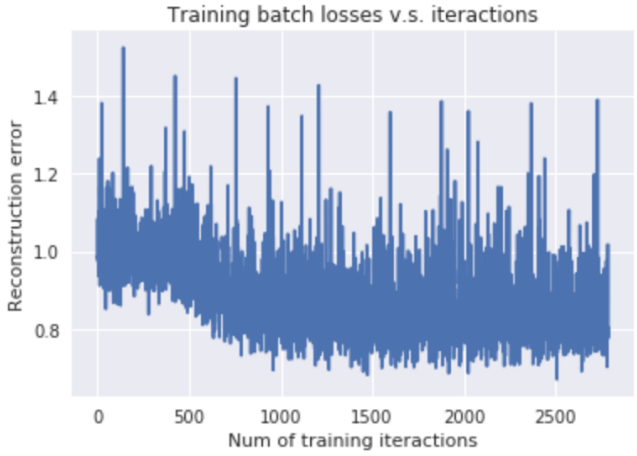
$$
\begin{aligned}
p(\mathbf{x}) &= \sum_{\mathbf{h}\in\{0,1\}^H} p(\mathbf{x},\mathbf{h}) = \sum_{\mathbf{h}\in\{0,1\}^H} \exp(-E(\mathbf{x},\mathbf{h}))/Z \\
&= \exp\left(\mathbf{c}^\top\mathbf{x} + \sum_{j=1}^{H} \log(1+\exp(b_j + \mathbf{W}_j.\mathbf{x}))\right)/Z \\
&= \exp(-F(\mathbf{x}))/Z
\end{aligned}
$$

free energy

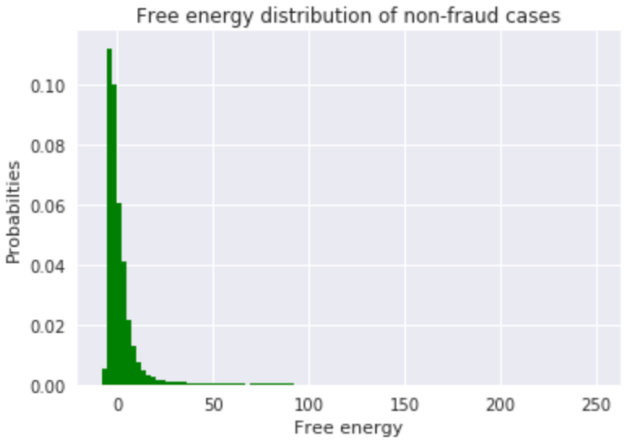
Higher the value of the free energy, the higher is the chance of x being a fraud.

To tune the hyper-parameters, we stick to the data-driven approach i.e. we start with hidden layer with a smaller dimension than the input layer and a small learning rate of the order 0.001.

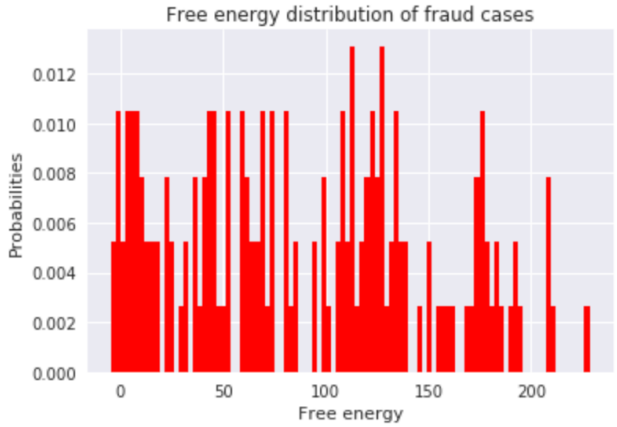
For training and validation, we split the data by time as 50-50 and train our model on the training set. After which we use the Free Energy of the validation set and visualize the results for fraud and non-fraudulent samples.



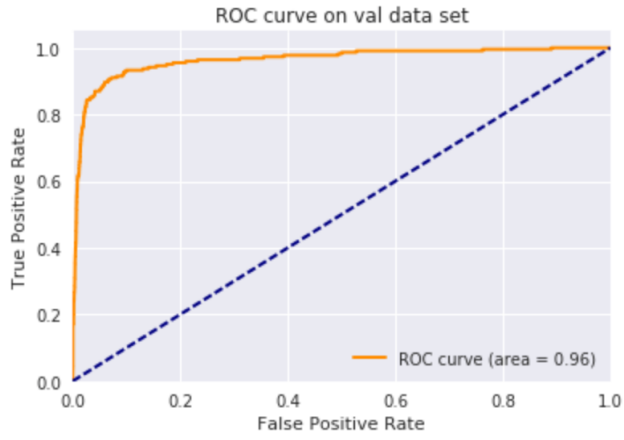
Like in the Autoencoder, we use z-score normalization to standardize the data. After that we have used Gaussian-Bernoulli RBM to fit the data with the model (since the hidden layer is binary-valued).



Next we check the Free Energy released by fraudulent and non-fraudulent transactions and compare them. It is very obvious that the free energy release from fraudulent data will be very high compared to non-fraudulent transactions.



Calculating the AUC score which is nothing but the area under the ROC curve, we get a score of around 0.967 which is almost equivalent to what we got in case of the Autoencoder.



IV. CONCLUSION

Thus we have successfully trained two unsupervised models to classify fraudulent v/s non-fraudulent transactions using Autoencoder and RBM. The good AUC scores suggest that our model is good and can produce correct results. There is still a lot to explore with the data set, we could preprocess data with accurate labels and test the accuracy of supervised learning algorithms on the same or use other variants of unsupervised algorithms to compare accuracies with Autoencoder and RBM.

V. REFERENCES

[1] Credit Card Fraud Detection with Unsupervised Algorithms Journal of Advances in Information Technology Vol. 7, No. 1, February 2016
[2] Data Set
https://www.kaggle.com/dalpozz/creditcardfraud
[3] https://weiminwang.blog/2017/06/23/credit-card-fraud-detection-using-auto-encoder-in-tensorflow-2
[4] https://weiminwang.blog/2017/08/05/credit-card-fraud-detection-2-using-restricted-boltzmann-machine-in-tensorflow/
[5]https://en.wikipedia.org/wiki/Credit_card_fraud
[6]
https://www.youtube.com/watch?v=FJ0z3Ubagt4
[7]
https://deeplearning4j.org/restrictedboltzmannmachine
[8] Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi. Calibrating Probability with Undersampling for Unbalanced Classification. In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015
[9] https://github.com/aaxwaz/Fraud-detection-using-deep-learning