Fraud Detection using AutoEncoders

Hesam Damghanian
SBU, Computer Science Dept.
h.damghanian@mail.sbu.ac.ir

***Abstract*— Autoencoders are an unsupervised neural network architecture that are consisted of an encoder and a decoder. The encoder part converts input data to a space with much lower dimensions called latent space, then the decoder part decodes samples from the latent space to a space with same dimensions as the input to reconstruct a sample same as the input sample. By training an autoencoder to reconstruct samples like the input data (normal transactions) and measuring the MSE between the input data and the reconstructed samples we can detect the fraud transaction from the normal ones based on this MSE error.**

Keywords—Autoencoders, Fraud Detection

I. Dataset

The given dataset is consisted two tables, one called transactions and the other called identity. The transaction table represents all transaction possible with 394 features, making a total of 543432 rows, and the identity table represents the processed transactions with 144233 rows total and 41 features. Both tables had numerical and categorical data which we see the method used in encoding them in the next section.

II. Preprocessing

All the categorical columns been encoded with categorical encoding scheme which simply assigns a unique number to every unique value in the column and is consistent within all the data points and -1 assigned for the entries with null values. Numerical columns were fitted with minmax scalar which maps the min value to zero and max value to one and all other values are scaled into this interval and null values were replaced with the mean value of the columns. Finally, I joined these tables based on the identity table together resulting the final table to have 433 columns along with 144233 rows.

Table 1, table 2 show details related to the transaction and identity table, respectively.

Table 1. Transaction features

Transaction columns	Description
TransactionDT	Time difference with respect to a starting time
TransactinoAMT	Amount of the transaction in USD
ProductCD	Product code
Card1-card6	Card related data such as type, bank, etc.
Addr	Region of the transaction.
Dist	Distance between the locations.
P/R emaildomain	Email domain related to customer and reseller.
C1-C14	Unknown features.
D1-D15	Time differences between transactions…
M1-M9	Relations between features
Vxxx	Extracted features like ranking, quantity, etc.

Table 2. Identity features

Identity columns	description
Unknown feature names	Connection and digital signature related info

Dimension Reductions

The joined dataset has 433 columns which makes the input space too large and more importantly sparse. I reduced this input dimension (433) to 128, 100, 64, 32, 16 with PCA and after grid search on these dimensions, it turned out that input space with 64 dimensions does better than the other input dimensions. Next, I divided the dataset into anomaly and normal datasets.

III. Experiments

Autoencoder Architecture

Figure below shows overall architecture along the experiments and number of layers within encoder and decoder part has been a variable in the hyperparameter tuning. First layer of encoder and last layer of the decoder have the same number of units equal to the input dimension. All layers are fully connected dense layers.

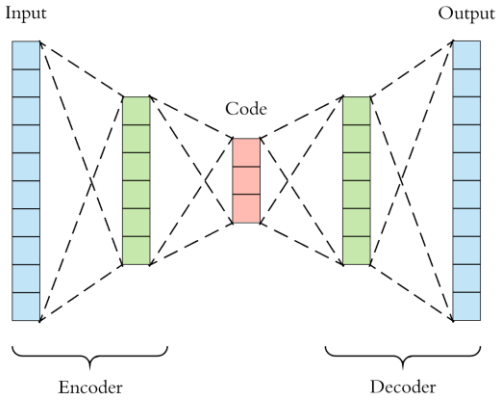


Figure 1 Autoencoder Architecture

Another experiment, the second notebook, also was done with higher units in the first layer of the encoder and last layer of the decoder to fully utilize the networks capacity to learn the input features (input and output layer have the previous dimension). Which yielded better results in classifying the heldout test set 75%, model itself and its weights are available in attachments. The test dataset classification results are also attached.

Hyperparameter Tuning

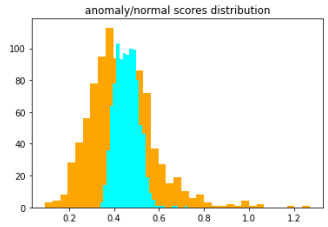
To gain better results, I conducted a relatively large grid search over following domains: depth of the autoencoder, dimensions of the input data, batch size, loss function, optimizer, and activation function. Each time model was set with a combination of these domains and the evaluated based on this formula:

$$\text{anomalyScore} = \text{RMSE}(\text{anomaly\_pred}, \text{anomaly})$$

$$\text{normalScore} = \text{RMSE}(\text{normal\_pred}, \text{normal})$$

$$|\ \text{normalScore} - \text{anomalyScore}\ | \ / \ \max(\text{normalScore}, \text{anomalyScore})$$

Which scores models higher that better classify anomalies from normal data. Figure below depicts the distribution of normal and fraud data scores, given the trained autoencoder only on normal data.



Unfortunately, tensorboard was not able to monitor the custom score that I defined, making 3 hours of training (192 variations), worth nothing :). Finally, I picked some values for variables eyeballing the log from output.

IV. Results

The train dataset has been split into test, train, and validation dataset. (0.2 test, 0.2 validation). The model was trained on train dataset and obviously validation set has been used for validation along training phase. After training completed, the heldout set, test set, was used for evaluating the autoencoders ability to classify. Classification was based on whether the reconstruction error for a given sample was closer to normal portion of training reconstruction error, to classify as normal, or the fraud portion. The autoencoder in the second notebook was able to classify 75% of the heldout sample correctly.