MFCDA Challenge 2017 Report

Guy Danielli

## Tools

Python 3.6 (Pycharm IDE) using Keras with Theano GPU as beckend.

## Methodology

The data we have contains shell commands histories of 40 different users, divided into 150 sessions of sequel commands per user. A session can be labeled as a *normal* session or as a *frauded* one. Normal sessions are labeled with 0 whilst frauded ones are labeled with 1.

I used the given first 50 sessions of every user for training and the remaining 100 sessions of the first 10 users for testing and evaluation.

I used a single classifier for each user. The classifier was trained with its user’s data as ‘0’ labeled data and other users’ data as ‘1’ labeled data. The input of the model is a vector that represent the number of appearances of features in the session. The features I used are ngrams that appear in the users’ data (for each classifier the input shape is identical).

I tried ngrams of different sizes (1 <= n <= 5). Bigrams and trigrams seemed to work best, whereas bigrams are slightly better than trigram and produce significantly less features (7000 compared to 18000), which results in much smaller models.

For the model I used a fully-connected neural network with the number of features as number of input units, a single hidden layer with *[ratio \* input\_dimension]* units, and and output layer with a single unit. For the hidden layer I tried using *relu* and *linear* activations. A linear activation seems to be superior to relu in this case. The output neuron’s activation was set to either one of sigmoid or hard-sigmoid, which seem to work well. The ration I used most of the time was set to be 0.002.

Since the model produces a number between 0 and 1 as a prediction, I had to choose a method of labeling samples based on their prediction. The first method I tried was to label any sample with prediction value > 0.5 as ‘1’ and the remaining samples were labeled as ‘0’. The next thing I tried is to ‘tighten the rope’, by incrementing the threshold by small steps and choosing the one that produced maximum score over the test dataset (this was applied at each run), calculated as the mean score over the 10 first users, whereas each user’s score is calculated using the following formula: *score =* *9\*tp + tn*. Then I decided a better way is to find the highest prediction value for the current user, *pmax*. Each prediction *p* that satisfies the condition: *pmax – p < epsilon*, was labeled ‘1’. This allows me to simulate a different threshold for each classifier, based on the classifier’s own confidence. Choosing the 10 samples with highest prediction value was also considered, though it produced low scores. After seeing that my highest score produced many false-positives for some users, I decided to improve it by combining both methods: samples that produced predictions with *threshold* distance from *pmax* and are within the 26 samples with highest prediction values are labeled as ‘1’. All other samples were labeled ‘0’.

In earlier experiments, I tried using also One Class SVMs and Auto Encoder ANN as models in order to apply one-class classifications, but they produced lower scores.

Another improvement I applied was to change the distribution of the labels within the training data. The first method for this was to use less data from other users’ sessions in the training data for one’s classifier. Using that method, though seemed to be working well sometimes, harmed the diversity of the data and limited the number of different examples seen by a single classifier. The better method was to simply clone the classifier’s own user’s sessions and use less training epochs. The later method is superior to the former only by little much score-wise, but another reason to use it is that the former method randomly chooses samples for learning and might produce different results at each run. I found that the number of clones that worked best for me moved between 7 and 9.

The predicted score was usually a higher than the actual score gained from a prediction sheet, though higher scores on the testing data did result in an increase of the actual score.