Write Up: "Limited Feature Fraud-Detection DNNs Using Sigmoid & Leaky ReLu"

Liam R. Watson

The key component of my prediction engine was an in-house written Deep Learning library supporting many high-level functions including "genetic" learning algorithms, time-series predictions etc. as well as extremely tunable and focused hyperparameters with a self-adjusting learning rate to accommodate maximal training, while also preventing the infamous "exploding gradient" we're all familiar with. I am the sole contributor to the library and have developed it from the ground-up in C and Python, with the former being used for mathematical calculations (performance reasons) and the latter supporting the rest of the Machine Learning stack.

For this project, the data available was large in quantity, and highly relevant to the task of fraud-detection systems for banking. This initially led me to be generous in selecting features for the *Data Exploration* phase. However, given my own personal computational limits and the time allotted, I quickly realised the extensive set of parameters was going to consume more-than-practical amounts of time (and not realistic in a production-environment such as Credit-transaction processing).

As a result, the feature-set was reduced to 40 input parameters (see ./Main Learner/DATA/desired_headers.txt), which were chosen very carefully using inspiration from research papers on ML for Fraud-Detection:

- https://doi.org/10.1016/j.procs.2020.01.057
- https://www.researchgate.net/publication/336800562_Credit_Card_Fraud_Detection_using_Machine_Learning_and_Data_Science

Included in this are AMOUNT, all FLAG headers, and COUNT 7DAY headers. This was extracted, and then parsed and formatted so that the AMOUNT was provided raw, while COUNT 7DAY values were a proportion of the total for that cardholder.

Data was also split 2:1 for Training:Validation, and additionally only a subset with equal FRAUD and NOT FRAUD cases was chosen; this was simply because the learning algorithm was favoring accepting **all** transactions as a means to gain accuracy – clever, but not what we wanted. Splitting it this way forced it toward an accurate gradient descent.

Hidden layers were sized in a "bubble" topology with the largest containing 36 neurons, and each calculated using *Leaky Rectified Linear Unit*.

There are 2 output parameters, each of which calculated using *Sigmoid*, and representing the probability of NOT FRAUD and FRAUD, respectively. The PREDICTION was then chosen to be the greatest of the two, while PROBABILITY was chosen to be [max(output values)-min(output values)].

After roughly 600 Epochs of training/cross-validation, the model was near-maximally trained (the balance between under and over -trained); it's possible that a tiny bit more performance could be achieved with long training.

Performance (validation set):

- FRAUD cases detected: 60% - 80% (depending on confidence level threshold)
- NON-FRAUD cases mistaken to be FRAUD: 5% - 8% (depending on confidence level threshold)

Finally, we can see that performance is rather high given the size of the network, so we've detected a significant portion of FRAUD transactions with minimal accidental detections. This is crucial since a business such as Scotiabank needs to balance customer/transaction growth with counteracting fraudsters. This model is only a prototype, so in a production environment using CUDA C/C++ (in my library as well) on GPU's, the performance could be increase by several dozen more percentage points, while keeping the code lean enough to run day-to-day.

Using this model, my recommendation for any business with the means to do so would be to implement high-performance DNN's with limited feature-sets, constant accuracy monitoring and consistent updated learning. This would support a lean, fast detection system that will dampen FRAUD cases, increase consumer uptake and reduce costs for the business, while increasing profits for a negligible upfront cost. Meanwhile, the system will also increase in performance over time.

Footnote: I also explored the use of Formal Concept Lattices (FCL's) which I have experience developing; this would require condensing the data into a matrix by using a threshold vector. If DNN's are too slow, FCL's would be the way to go given it's Glass-box (explainable) and incredibly fast.

References:

Dornadula, Vaishnavi Nath, and S Geetha. "Credit Card Fraud Detection Using Machine Learning Algorithms." *Procedia Computer Science*, vol. 165, 2019, pp. 631–641., https://doi.org/10.1016/j.procs.2020.01.057.

S P, Maniraj & Saini, Aditya & Ahmed, Shadab & Sarkar, Swarna. (2019). Credit Card Fraud Detection using Machine Learning and Data Science. International Journal of Engineering Research and. 08. 10.17577/IJERTV8IS090031.