Paul Lintilhac

Consensys – Final Round Exercise

For this exercise, I decided to use RStudio for my analysis, using the following packages either in my preliminary analysis or the models themselves: data.table, Matrix, lattice, expm, rARPACK, pmclust, foreach, and doParalel. In particular, since there were over 3 million transactions and I needed to perform matrix operations on a 230,000x230,000 sparse matrix (or 40 billion entries), I used rARPACK which implements many standard matrix operations for very large, sparse matrices. I also used pmclust, doParallel and foreach to parallelize both the cross-validation process and the clustering algorithm itself.

**Characterizing the data:**

Characterization of all transactions:

First off, I will note that out of the roughly 3 million transactions provided, there are only about 230,000 individual “to” or “from” addresses, meaning that there are roughly 10 transactions per address in the data set provided.

While assessing the risk of a given address, we need to consider not only address that transacted to that address, but also from that address. I am assuming in this analysis that a transaction from B to A as equivalent as a transaction from A to B. Said mathematically, we are characterizing the transactions as an undirected graph, where each edge is characterized by an unordered pair of addresses.

Interestingly, it turns out that very few of the addresses are connected in a bi-directional way anyways. Only about 7000 pairs of accounts are connected in a bi-directional way, and these bi-directional relationships are constituted by about 5000 unique accounts.

Whereas the average degree of the nodes in this graph is 3.6, the average degree of nodes with a bidirectional edge is only 1.3 – indicating that these nodes likely represent two accounts that are very closely related, and transact almost exclusively with one another.

Characterization of the bad transactions:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | gas | gas.price | Gas.used | value | Gas.used.block |
| Bad txs | 173268 | 51532042413 | 149609.9 | 8967124 | 1198083 |
| All txs | 125032.6 | 27243941280 | 38539.59 | 36314985 | 1290476 |

Characterization of the blocks associated with bad transactions:

I also analyzed differences between blocks containing transactions with bad accounts versus average blocks. Interestingly, blocks with bad transactions had an average gas.used of 1217368, as opposed to the average gas used in a block of 282889.8. This shows that blocks containing bad transactions have an order of magnitude higher rate of gas used in the entire block.

In addition, it should be noted that there are 16 blocks which have the same block number. Since they also had the same hash, I assumed they were simply duplicates in the data, rather than a split in the blockchain, so I simply de-duped them.

**Question 2:**

I will address question 2 using the concept of the Laplacian Matrix from graph theory. We can consider the risk model to be a diffusion model, where the risk is initially concentrated at the known bad nodes, and then it diffuses outwards to any adjacent nodes. As discussed above, we are considering the graph to be undirected, and that edges in either direction count towards sums and flags equally. Specifically, when looking at the value or number of transactions between two addresses I am looking at the sum of these figures in each direction for that edge, while the flag representing an edge between those two addresses is given by the maximum of the flag in each direction (i.e. two addresses are connected if they have a transaction in either direction).

Definitions and background:

We consider a random vector of probabilities v(t) in Rn­­, where n is the number of unique addresses in our transaction data set. Let G be the graph representation of these addresses, with an edge representing a transaction between two addresses in either direction. Suppose D is the diagonal matrix of degrees of each address node, and that A is the adjacency matrix. Then the Laplacian matrix L is defined as A-D.

The diffusion model starts with a binary vector of probabilities where every entry is 0 except for the addresses known to be involved in attacks, which have values of 1. Then if we consider the likelihood of of being an accomplice to one of the bad notes to be uniformly distributed among all adjacent nodes, we can think of the evolution of the system as

As a contextual side-note, if we were to replace L with the continuous operator of the Laplacian (a kind of second derivative), this would simply be the heat equation, where the analogous initial condition would be a point mass of heat introduced into the system. The solution to this equation looks is:

Note that I am using the matrix exponential here, which can be defined using the matrix form of the Taylor Series. I could not find any packages that could compute the matrix exponential for a matrix of this size, so I instead took a taylor approximation of the matrix exponential. Taking t as a constant, we can absorb it into the constant k and we have the formula

The result is a vector that assigns a probability to all nodes of being involved in the attack with this diffusion model. Of course the actual attacker addresses will always have the highest score, and nodes that are close to one or more attacker node will have non-zero scores. And nodes that are disconnected from these nodes entirely will always have a score of 0. Note that the second order approximation involves the product L\*L, i.e. it accounts for not only neighbors of bad accounts but neighbors of neighbors, with a lesser weight due to the presence of the term which is very small. Higher order approximations will involve nodes an arbitrary distance from the attacker nodes with weights given by the corresponding Taylor approximation.

As a generalization of his model, we may not want our diffusion probabilities or “strengths” to be uniformly weighted among all adjacent nodes. Instead, we may want to count multiple transactions on a given pair more than once, or we may want to weight the edge based on the total gas used in all transactions, or the total value. In this case, we would replace the adjacency matrix A with a matrix with these quantities in each entry (rather than a simple binary “edge” flag) and form a weighted adjacency matrix A\*, and then instead of the degree matrix, we subtract from A\* a diagonal matrix D\* with its entries equal to the column sums of A\*, in order to get the weighted Laplacian L\*=A\*-D\*. The diffusion equation using these different diffusion models then has the same form. If we wanted to use a combination of these models, we would have to do a weighted average:

Where k is the number of different weighted laplacians we are using, and is the corresponding weight of that model, which again would be free parameters of the model to be optimized.

Summary:

For this problem I chose k=.01 and flagged any accounts with a probability score as “bad” accounts. I chose to use the second order Taylor approximation for the diffusion, and stuck with the unweighted Laplacian matrix for my analysis. This resulted in a total of 52 accounts being flagged, with roughly 100,000 associated transactions, or roughly 3% of the data set. These labels based on the diffusion model are used to test the anomaly detection model in the next problem.

**Question 2:**

I approached this problem by using a Gaussian Mixture Model (GMM) in order to perform unsupervised clustering on 5 subsets of 10,000 transactions each. Ideally I would be able to use more transactions for the analysis, but unfortunately I was constrained with computer power. I was able to get an order of magnitude increase in speed by running the cross validation in parallel over 8 cores and using the parallelized version of the GMM clustering algorithm provided by the package “pmclust”. But ideally there would be a fully parallelized version of this that could be run on an AWS cluster with spark using the package rSpark or pyspark for python.

From this analysis we can recover the probability of a point belonging to its assigned cluster. Under the assumption that the flagged transactions are “abnormal” in some general sense, we can test the model by comparing the average likelihood of all transactions with the average likelihood of transactions related to the accounts flagged in the previous part.

In addition, we may wish to consider the possibility that the bad transactions are actually being captured quite well within a cluster of other transactions. We would be able to see this by noticing a change in the distribution of bad transactions over the clusters as compared to the distribution of all transactions over the clusters.

Of course, it is also possible that neither of these assumptions is true, because the bad transactions simply don’t stand out in any way.

Data:

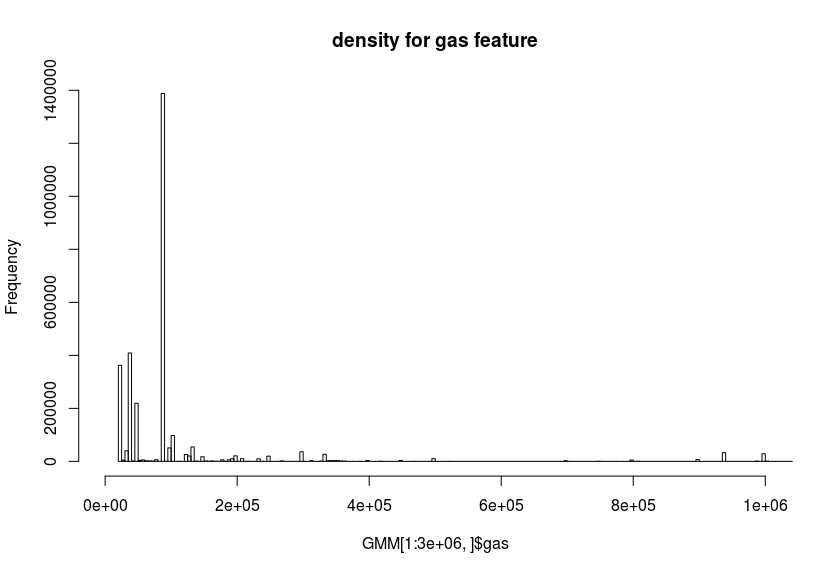
By merging together the transaction and block data set, I compiled a dataset that has the following quantitative information about each transaction:

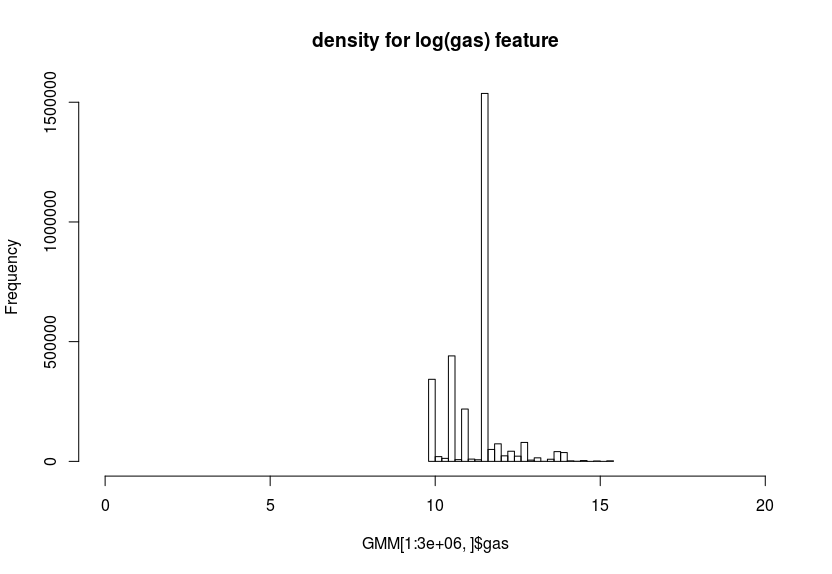
* Tx-related data: gas,gas.used.tx,gas.price,value
* Block-related data: gas.used.block
* I joined in some data from the previous graphical analysis, including the degree of the “from” account and the degree of the “to” account in the graph.

I did not include the timestamps of the block or the transaction as features in this simplified model, since the time variable should be treated separately, and would require more time to be taken than this analysis allows.

The maximum correlation of all 7 features used was .45, meaning that there was not a significant amount of collinearity in the features, which would violate the assumptions of the GMM model. But since there was no real colinearity, there was no need to eliminate any of the features.

Finally, I noticed that many of the quantitative features had a distribution that was far from normal, which would cause the GMM model to perform badly. However, I did notice that these variables had something that looked much closer to a lognormal distribution, wth no values below zero, and a long right tail. Therefore I transformed the variables with the log() function, and the result was that these variables now seemed approximately normal. An example of this transformation for the gas feature is shown below:





Summary:

I performed this clustering with 4 disjoint subsets of the data (I chose k=4 because I have 8 cores, so it is optimal for parallelization for it to be a multiple of 4), each with 10000 transactions, and using K=4 for the number of independent Gaussian models used in the mixture. I then compared the likelihood scores for bad transactions to the likelihood for all transactions for each test. Then I looked at the difference in distribution over the four clusters for the bad transactions vs all transactions. My results are summarized below:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| bad | fold | C1 | C2 | C3 | C4 | prob | uncertainty |
| bad | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| good | 1 | 0.09 | 0.536 | 0.159 | 0.214 | 0.998695 | 0.001305 |
| bad | 2 | NA | NA | NA | NA | NA | NA |
| good | 2 | 0.057 | 0.522 | 0.127 | 0.294 | 0.99717 | 0.00283 |
| bad | 3 | NA | NA | NA | NA | NA | NA |
| good | 3 | 0.276 | 0.05 | 0.506 | 0.168 | 0.999261 | 0.000739 |
| bad | 4 | NA | NA | NA | NA | 1 | 0 |
| good | 4 | 0.493 | 0.036 | 0.304 | 0.167 | 0.998657 | 0.001343 |

Notice that in this trial, there were only bad transactions in the first fold of the cross-validation. It appears from these results that in fact the first hypothesis – that the “bad” transactions should have a lower likelihood score – is false. It turns out that in fact these transactions have a higher likelihood score and lower uncertainty than the average transaction.

However, this could still be because the second hypothesis is true – namely that these anomalous transactions are actually being captured quite well in one of the four clusters. Indeed, this appears to be the case, as we can see from the fact that in the first fold, 100% of the bad transactions are in cluster 1, whereas only 9% of transactions in the total sample are in this cluster. This suggest that the first cluster represents attack risk.

Below is a table showing the means of cluster 1 vs cluster 2:

|  |  |  |
| --- | --- | --- |
|  | Cluster 1 | Cluster 2 |
| log(gas) | 12.729 | 10.776 |
| log(gas.price | 23.972 | 23.890 |
| log(gas.used.tx) | 11.221 | 9.958 |
| log(value) | -39.536 | 12.732 |
| log(gas.used.block) | 13.466 | 13.425 |
| from.degree | 32.057 | 2230.278 |
| to.degree | 1.812 | 147.341 |

We can see from this that the most outstanding discrepancy between these clusters is that cluster 1, which is associated with the bad transactions, has a significantly lower average value (almost 0), a significantly lower from.degree, and a significantly lower to.degree. This may go along with our intuition that these transactions are accounts being created by the SUICIDE or CREATE opcodes which have very low value, and can be produced in very high numbers. Note that the average from.degree and to.degree are much lower than the average for accounts here. Of particular interest is the fact that the average to.degree is lower than the average for all transactions in the parent data a set, 3.6. This could be because these accounts were created and then left inactive, or interacted with only one other account. Note that these average degrees disagree with the numbers provided above, because they are averages over transactions, not accounts (so the accounts with high degree are potentially being counted many times).

**Question 4:**

* To improve on the above modeling techniques, we could try to combine the diffusion model with the mixed Gaussian model by instead running a supervised learning algorithm using the extrapolated “bad” transactions generated by the diffusion model as labels. This would not be anomaly detection per se, but rather classification. There are many algorithms that would be well-suited for this, such as SVM or boosting, which work very well as binary classifiers over numeric vectors. This would likely yield better results in terms of the ability to predict bad transactions, but it did not seem like the main prompt here.
* My anomaly detection model was created on the transaction level. It would be good to have an account-level model as well, though it would be important to retain aggregate statistics for the transactions associated with that account, just as account level information (graph information) was merged into the transaction-level dataset.
* We could also incorporate the EthOn ontology framework into either the Gaussian anomaly detection model or the fraud diffusion model.
  + For the GMM, since the distribution of the features such as gas.used would be different between two contracts on the one hand and a contract and an addresses on the other, we would need to maintain different models for these two different types of transactions.
  + For the diffusion model, we could consider different rates of diffusion depending on what kind of transaction it is, using the EthOn ontology (a could be different when we are talking about a transaction between two contracts vs an address and a contract)
* In the diffusion model we could allow risk to be diffused along the graph at a different rate from a “from” node to a “to” node than it is from a “to” node to a “from” node. This would be a pretty natural generalization
* Introduce a time component into the models. This could take several forms, but could start very simply by introducing a seasonal or trend component into the model parameters during the training phase.
* Consider using sophisticated new online graphical learning algorithms like the SHAZOO algorithm that can approximate the graph using spanning trees and can have major performance benefits.