Non-Negative Paratuck2 Tensor Decomposition Combined to LSTM Network for Smart Contracts Profiling

Jeremy Charlier* and Radu State
SEDAN, University of Luxembourg, 29 Avenue J.F Kennedy, L-1855, Luxembourg, Luxembourg

Introduction

In the next months, public institutions and governments will certainly start regulating the non-regulated activities of cryptocurrencies such as Bitcoin or Ether. Some governments already claimed they were investigating cryptocurrencies activities [1-3]. These regulations will probably introduce new sets of rules and ask for more transparency among the blockchain players. As a result, financial products would probably require key information document to advise potential investors of the risk of these investments. Ethereum, with already more than one million accounts, is one of the major platforms for smart contracts relying on Ether cryptocurrency for its existence. Still, the platform supports very few documentation for its existence. Therefore, smart contracts programs. These are self-executing blockchain contracts. Due to their high volume of transactions, analyzing their behavior is very challenging. We address this challenge in our paper.

Methods: We develop for this purpose an innovative approach based on the non-negative tensor decomposition Paratuck2 combined with long short-term memory. The objective is to assess if predictive analysis can forecast smart contracts activities over time. Three statistical tests are performed on the predictive analytics, the mean absolute percentage error, the mean directional accuracy and the Jaccard distance.

Results: Among dozens of GB of transactions, the Paratuck2 tensor decomposition allows asymmetric modeling of the smart contracts. Furthermore, it highlights time dependent latent groups. The latent activities are modeled by the long short term memory network for predictive analytics. The highly accurate predictions underline the accuracy of the method and show that blockchain activities are not pure randomness.

Conclusion: Herein, we are able to detect the most active contracts, and predict their behavior. In the context of future regulations, our approach opens new perspective for monitoring blockchain activities.
The transpose matrix of $A \in \mathbb{R}^{i \times j}$ is denoted by $A^T$.

The Moore-Penrose inverse of a matrix $A \in \mathbb{R}^{i \times j}$ is denoted by $A^+$.

$X$ is called a $n$-way tensor if $X$ is a $n$-th multidimensional array. It is expressed by $X \in \mathbb{R}^{i_1 \times i_2 \times \ldots \times i_n}$.

The square root of the sum of all tensor entries squared of the tensor $X$ defines its norm.

$$\|X\| = \sqrt{\sum_{j=1}^{i_1} \sum_{j=2}^{i_2} \ldots \sum_{j=n}^{i_n} x_{j_1, j_2, \ldots, j_n}^2}$$

(1)

The rank-$R$ of a tensor $X \in \mathbb{R}^{i_1 \times i_2 \times \ldots \times i_n}$ is the number of linear components that could fit $X$ exactly such that

$$X = \sum_{r=1}^{R} a_r^{(1)} \otimes a_r^{(2)} \otimes \ldots \otimes a_r^{(N)}$$

(2)

with the symbol $\otimes$ representing the vector outer product.

The Kronecker product between two matrices $A \in \mathbb{R}^{i \times j}$ and $B \in \mathbb{R}^{k \times l}$, denoted by $A \otimes B$, results in a matrix $C \in \mathbb{R}^{ik \times jl}$. It is the column-wise Kronecker product.

$$C = A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \ldots & a_{1j}B \\ a_{21}B & a_{22}B & \ldots & a_{2j}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{ij}B & a_{i2}B & \ldots & a_{ij}B \end{bmatrix}$$

(3)

The Khatri-Rao product between two matrices $A \in \mathbb{R}^{i \times k}$ and $B \in \mathbb{R}^{j \times k}$, denoted by $A \odot B$, results in a matrix $C$ of size $\mathbb{R}^{ij \times k}$. It is the column-wise Kronecker product.

$$C = A \odot B = \begin{bmatrix} a_1b_1 & a_1b_2 & \ldots & a_1b_k \\ a_2b_1 & a_2b_2 & \ldots & a_2b_k \\ \vdots & \vdots & \ddots & \vdots \\ a_ib_k \end{bmatrix}$$

(4)

**Non-Negative Paratuck2**

The Paratuck2 decomposition, [9], is well suited for the analysis of intrinsically asymmetric relationships between two different sets of objects. It represents a tensor $X \in \mathbb{R}^{i_1 \times j_1 \times k_1}$ as a product of matrices and tensors.

$$X = \sum_{k=1}^{K} A_k^4 H D_k^2 B_k^T$$

(5)

A, $H$ and $B$ are matrices of size $\mathbb{R}^{i_1 \times p}$, $\mathbb{R}^{q \times j_1}$, and $\mathbb{R}^{k_1 \times q}$, respectively. The matrices $D_k^2 \in \mathbb{R}^{p \times p}$ and $D_k^4 \in \mathbb{R}^{q \times q}$, $\forall k \in \{1, \ldots, K\}$ are the slices of the tensors $D_k^4 \in \mathbb{R}^{p \times p \times q}$ and $D_k^2 \in \mathbb{R}^{q \times q \times k_1}$. The latent factors $p$ and $q$ are related to the rank of each object set as illustrated in figure 1. The columns of the matrices $A$ and $B$ represent respectively the latent factors $p$ and $q$, the matrix $H$ describes the asymmetry between the $p$ latent factors and the $q$ latent factors. Finally, the tensors $D_k^4$ and $D_k^2$ represent the degree of participation, also called strength, for each of the latent factors, respectively $p$ and $q$, according to the third dimension.

To achieve the computation of the Paratuck2 decomposition, the following minimization equation has to be solved

$$\min_{\hat{X}} \| X - \hat{X} \|$$

(6)

with $\hat{X}$ the approximate tensor described by the decomposition and $X$ the original tensor.

To solve equation 6, the Alternating Least Squares (ALS) method is used as presented by Bro in [10]. All of the matrices and the tensors are updated iteratively. To simplify the resolution explanation, we consider one level $k$ of $K$, the third dimension of the tensor.

To update $A$, equation 5 is rearranged such that

$$X_k = AF_k^+$$

with $F_k = D_k^4 HD_k^2 B_k^T$.

(7)

The simultaneous least square solution for all $k$ leads to

$$A = X(F_1^+)^T$$

(8)

To update $D_k^4$, equation 5 is rearranged such that

$$X_k = AD_k^4 F_k^+$$

with $F_k = BD_k^2 H_1^T$.

(9)

The matrix $H_1$ is a diagonal matrix which lead to the below resolution.

$$D_k^4 = (F_k \odot A) X_k$$

(10)

The notation $(k, :)$ represents the $k$-th row of $D_k^4$.

To update $H$, equation 5 is rearranged such that

$$X_k = (BD_k^2 \otimes AD_k^4) h$$

with $X_k = \text{vec}(X_k)$

(11)

which brings the solution

$$h = Z^T x$$

with $Z = \begin{bmatrix} BD_k^2 \otimes AD_k^4 \\ BD_{k+1}^2 \otimes AD_{k+1}^4 \\ \vdots \\ BD_K^2 \otimes AD_K^4 \end{bmatrix}$

(12)

---

Figure 1: Paratuck2 decomposition of a three-way tensor with dimension notations.
To update $B$ and $D^b$, the methodology presented for the update of $A$ and $D^a$ is applied respectively.

In the experiments, we use the non-negative Paratuck2 decomposition leveraging the non-negative matrix factorization presented by Lee and Seung in [19]. The matrices $A$, $B$ and $H$, and the tensors $D^b$ and $D^g$ are computed according to the following multiplicative update rule.

\[
\begin{align*}
    a_{ip} &\leftarrow a_{ip} \frac{[XF^T]_{ip}}{[A(FF^T)]_{ip}}, F = D^a HD^b B^T \\
    d_{pp}^a &\leftarrow d_{pp}^a \frac{[Z^T X]_{pp}}{[D^a(ZZ^T)]_{pp}}, Z = (BD^b H^T) \odot A \\
    h_{pq} &\leftarrow h_{pq} \frac{[Z^T X]_{pq}}{[H(ZZ^T)]_{pq}}, Z = BD^b \odot AD^a \\
    d_{qq}^b &\leftarrow d_{qq}^b \frac{[XZ]_{qq}}{[D^b(Z^T Z)]_{qq}}, Z = B \odot (H^T D^a A^T)^T \\
    b_{qq} &\leftarrow b_{qq} \frac{[X^TF^T]_{qq}}{[B(FF^T)]_{qq}}, F = (AD^a HD^b)^T
\end{align*}
\]

with

\[
\begin{align*}
    X &= [X_1, X_2, \ldots, X_k] \\
    X &= \text{vec}(x)
\end{align*}
\]

The non-negative multiplicative update rule helps to better calibration of LSTM since it uses the elements of the tensor decomposition as a starting point. Hereinafter is the complete algorithm of the non-negative ALS Paratuck2 resolution.

**Algorithm 1** Non-Negative ALS Paratuck2 with $P$ and $Q$ latent components for a tensor $X$ of size $I \times J \times K$

1. procedure NN-PARATUCK2($X, P, Q$)
2. random initialization $A \in \mathbb{R}^{I \times P}$, $H \in \mathbb{R}^{P \times Q}$, $B \in \mathbb{R}^{J \times Q}$
3. set $D^a \in \mathbb{R}^{I \times P}$ and $D^b \in \mathbb{R}^{J \times Q}$ equal to 1 for $k = 1, \ldots, K$
4. $X = [X_1, X_2, \ldots, X_k]
5. x = \text{vec}(x)$
6. repeat:
7. \[ a_{ip} \leftarrow a_{ip} \frac{[XF^T]_{ip}}{[A(FF^T)]_{ip}}, F = D^a HD^b B^T \]
8. \[ d_{pp}^a \leftarrow d_{pp}^a \frac{[Z^T X]_{pp}}{[D^a(ZZ^T)]_{pp}}, Z = (BD^b H^T) \odot A \]
9. \[ h_{pq} \leftarrow h_{pq} \frac{[Z^T X]_{pq}}{[H(ZZ^T)]_{pq}}, Z = BD^b \odot AD^a \]
10. \[ d_{qq}^b \leftarrow d_{qq}^b \frac{[XZ]_{qq}}{[D^b(Z^T Z)]_{qq}}, Z = B \odot (H^T D^a A^T)^T \]
11. \[ b_{qq} \leftarrow b_{qq} \frac{[X^TF^T]_{qq}}{[B(FF^T)]_{qq}}, F = (AD^a HD^b)^T \]
12. until maximum number of iterations or stopping criteria satisfied

**Latent LSTM predictions**

Based on the notation of Sak et al. in [20], LSTM contains memory blocks in the recurrent hidden layer. Each memory block is connected to an input gate and an output gate. Similarly to RNN, the input gate plays the role of the input activation of the memory cells. The output gate is in charge of the flow of cell activations into the rest of the network. In addition, a forget gate is added to the memory block.

![Figure 2: Overview of a LSTM memory cell. In our model, the activation functions $g$ and $h$ are described by $tanh$, and $f$ is the forget gate.](image-url)
since Gers, Cummins and Schmidhuber presented it in [21]. The forget gate allows the reset of the cell’s memory depending on the information received through the input gate. If we consider the input sequence denoted by \( x \) such as \( x = (x_1, \ldots, x_T) \), the output sequence denoted by \( y \) such as \( y = (y_1, \ldots, y_T) \) for a sequence of events from \( t = 1 \) to \( t = T \). The mapping between \( x \) and \( y \) for all network unit activations within LSTM is described by the set of equations (15). The activation of the input gate is denoted by \( i_t \), the candidate value for the states of the memory cells by \( C_t \), the activation of the memory cells forget gates by \( f_t \), the memory cells new state by \( C_t \), the value of their output gates by \( o_t \) and the outputs of the output gates by \( h_t \).

\[
\begin{align*}
    i_t &= \sigma(W_{xi}x_t + U_{hi}h_{t-1} + b_i) \\
    C_t &= \tanh(W_{xc}x_t + U_{hc}h_{t-1} + b_c) \\
    f_t &= \sigma(W_{xf}x_t + U_{hf}h_{t-1} + b_f) \\
    C_t &= i_t \cdot C_t + f_t \cdot C_{t-1} \\
    o_t &= \sigma(W_{xo}x_t + U_{ho}h_{t-1} + W_CC_t + b_o) \\
    h_t &= o_t \cdot \tanh(C_t)
\end{align*}
\] (15)

In the set of equations 15 at time \( t \), \( x_t \) stands for the memory cell layer, \( W_i \) and \( U_i \) with \( k \in \{i, c, f, o\} \) for the weight matrices and \( b_k \) for the bias vectors. In the model used for the experiments, the activation of a cell’s output gate is independent of the memory cell’s state \( C_t \) such that \( V_0 = 0 \). The main advantage by fixing \( V_0 = 0 \) is the ability to perform faster computation, especially on large datasets.

With regards to figure 1, the tensors \( D^A \) and \( D^B \) collects data about the tensor factorization related to the third dimension, which is very often the time. It means that the evolution of each groups, or clusters, characterized by the latent factors \( P \) and \( Q \) of the TD contained in the tensors \( D^A \) and \( D^B \) can be modeled using LSTM. More precisely, LSTM is calibrated on the historical data of the tensors \( D^A \) and \( D^B \) to predict afterwards the future evolution of each \( P \) and \( Q \) groups contained in the tensors \( D^A \) and \( D^B \) as illustrated in figure 3.

Only the diagonals of the tensors \( D^m \) with \( m = \{A, B\} \) contain numbers. Therefore, the tensors \( D^m \in \mathbb{R}^{L \times K} \) can be reduced to a matrix, \( E \in \mathbb{R}^{L \times K} \). The notation \( L = \{P, Q\} \) denotes the latent factors of the Paratuck2 TD.

**Figure 3**: Overview of LSTM training and predictions on the tensor \( D^m \in \mathbb{R}^{L \times L \times K} \) with \( m = \{A, B\} \) and \( L = \{P, Q\} \).

**Figure 4**: Three way tensor containing Ether amount exchanged between different smart contracts.
Data contained in $E$ is then used to train LSTM before performing the predictions on an interval $\epsilon$ related to the third dimension $K$. The resulting matrix of size $R^{L \times (K + \epsilon)}$ gathers the historical data of each latent component $L$ as well as the predicted values. A new tensor denoted by $D_s$ of size $R_L \times L \times (K + \epsilon)$ is built. The methodology is applied on both tensors $D^A$ and $D^B$ for the same $\epsilon$. Consequently, the Paratuck2 TD is linked to historical data and predicted data.

### Results and Discussion

In this section, we apply our multidisciplinary tensor neural network approach, Paratuck2-LSTM, for Ethereum smart contracts profiling. The experiment is performed on a machine with 15 Intel Xeon E5-4650 v4 2.20 Ghz CPU cores and 80 GB of RAM. We have implemented in Python the algorithm for non-negative Paratuck2 decomposition combined with LSTM code available in [22].

### Application to Smart Contracts

Smart contracts activities have been extracted from the Ethereum platform. The data was collected starting 1 January 2016 and ending 1 July 2016. Through the collection process, different data types have been stored, such as the hash key, the sender accounts, the receiver accounts or the blockheights. For the considered six months period, more than 5 millions of transactions have been made. This accounts for an average of 26 transactions per sender account and 18 transactions per receiver account.

In the data set, some smart contracts only relate to one transaction, payment or reception. Such behavior is difficult to predict, and should be considered as unexpected behavior. Our aim is to predict future interactions based on exchanges that already happened. Consequently, only the 1% most active smart contracts have been kept in the training set for their regular activities. This represents a list of 100 smart contracts sending an Ether amount, and a list of 200 smart contracts receiving an Ether amount from the sender contracts.

The features extracted from the dataset are well suited for a tensor representation. Two tensors denoted by $X \in R^{I \times J \times K}$ are built from the Ethereum data. The first dimension, $I$, lists the sender accounts, the second dimension $J$, the receiver accounts and the third dimension, $K$, the time. For each tensor, the interaction between a sender and a receiver is represented by the amount of Ether exchanged at a time. The dense tensor is built based on figure 4. The size of the tensor is $R^{100 \times 200 \times 50}$. The tensor is decomposed to highlights the latent component over time. Then, LSTM latent predictions are performed.

As illustrated in figure 5, the information evolving over time is contained in the tensors $D^m$ with $m=\{A, B\}$. The matrix $A$ gathers static information regarding $P$ senders groups and the matrix $B$ static information regarding $Q$ receivers groups. The matrix $H$ contains the asymmetric information between the $P$ and the $Q$ latent factors which have been set to respectively to 20 and 30. As a result, the LSTM network is trained on $D^m$ for the sender and the receiver activities predictions.

![Figure 5: Paratuck2 decomposition applied to smart contracts profiling. The model training and predictions are applied on the tensors $D^A$ and $D^B$.](image)

![Figure 6: Paratuck2-LSTM applied to smart contracts profiling. This simulation highlights training and predictions on one latent component of the tensor $DA$.](image)
Predictions results

Figures 6 and 7 show the difference between the true experimental data and the predictions for one rank of the tensors $D_A$ and $D_B$. The LSTM predictions of smart contracts activities are close to the one observed in the tensor decomposition of the complete true dataset. It means LSTM is appropriate for the modeling of the smart contracts having regular exchanges. To further quantify the accuracy of LSTM predictions, statistical tests are performed. The mean absolute percentage error (MAPE) and the mean directional accuracy (MDA) are computed between the predictions and the true data set. A third measure, the Jaccard distance, is also evaluated. As a benchmark for LSTM predictions, the results are compared to the predictions performed by a Decision Tree (DT).

Tables 1 and 2 highlights similar MAPE results for both LSTM predictions and DT predictions. Differences are not significant. On the other hand, the MDA score is lot higher, around a factor 7, for LSTM predictions than for DT predictions. It means the LSTM predictions are able to better reproduce the variations observed in the smart contracts activities through time than the DT predictions. From these first statistical tests, we can observe that the LSTM model is able to reproduce the changes over time of smart contracts activities. It outperforms the decision tree benchmarking algorithm. In addition, the Jaccard distance is computed to underline the distribution divergence between the predictions and the true experimental data. In table 3, it can be observed that LSTM predictions are significantly closer to the true experimental distribution than DT predictions. All LSTM Jaccard distances are within the range 0.20 and 0.30 while the DT Jaccard
distances are between 0.57 and 0.69. LSTM Jaccard distances are between 2 to 3 times lower than the DT Jaccard distances. It confirms the MDA scores in tables 1 and 2.

From the highlighted results, the combined approach of Paratuck2-LSTM delivered good results, validated visually and statistically. It outperformed the DT benchmarking for predictive analytics on several statistical criteria including the MDA and the Jaccard distance.

Conclusion

We proposed in this paper a multi-disciplinary approach leveraging multidimensional linear algebra and neural networks for modeling the complex activities occurring on a certain type of blockchains. Our method combines Paratuck2 tensor decomposition and LSTM to predict behavior in relation to asymmetric data over time. The asymmetry is expressed within the tensor decomposition using two sets of latent factors related to two sets of objects. Our use case considered sender and receiver contracts of the Ethereum platform. Our approach allowed to detect common behaviors over time. Furthermore, it was able to predict accurate interactions and exchanges. We validated our results using statistical tests.

Although the method showed good results in terms of accuracy, it currently lacks the required scalability to be used on big data sets. This is due to the non-negative ALS update rule which is time and memory consuming. We plan to address this issue and develop additional resolution method to the Paratuck2 tensor decomposition using other iterative schemes. Last but not least, the better scalability of the method would help to increase the accuracy of the LSTM network as the training could be performed on longer time period and smaller time step discretization. We plan to address a particular use-case about fraud detection and detection of suspicious behavior over time.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgements

The authors would like to thank Beltran Borja Fiz Pontiveros for his help in Ethereum data manipulation.

References
