Proceedings of the 4th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/48/20241181

Enhancing predictive models for illicit activities in the Bitcoin transaction network using advanced graph analytical techniques

Yancheng Pan

School of Computer Science and Engineering, South China University of Technology, Guangzhou, 511400, China

201930343049@mail.scut.edu.cn

Abstract. The Elliptic dataset compiles a comprehensive history of Bitcoin transactions, integrating both anti-money laundering (AML) tags and distinct graph network features. Given the nature of the Bitcoin transaction network—a complex, weakly interconnected structure—leveraging graph analysis techniques for its study holds immense potential, especially in the realm of detecting illicit activities like hacking, drug trades, gambling, and more. A detailed examination of the Elliptic dataset, encompassing transaction amounts, frequencies, source and destination addresses, sheds light on the inherent structure and peculiarities of the Bitcoin transaction ecosystem. By conceptualizing this transactional landscape as a graph, a slew of analytical attributes emerge: node degree distribution, community architecture, centrality measures, and so forth. Such attributes pave the way for the creation of predictive models that can pinpoint and prognosticate potential unlawful trade actions. Several computational models have been employed on the Elliptic dataset, such as Logistic Regression (LR), Random Forest (RF), Multilayer Perceptrons (MLP), and Graph Convolutional Networks (GCN). The authors of this particular study delve into augmentations of the GCN model, juxtaposing the efficacy of the original GCN model against their enhanced algorithm within the context of the Elliptic dataset.

Keywords: Elliptic dataset, Graph Convolutional Networks, Cryptocurrency, Visualization.

1. Introduction

The meteoric rise of the Internet and the surge in e-commerce transactions have unfortunately provided fertile ground for fraudulent activities. These illicit endeavors lead to considerable economic losses, affecting both individuals and organizations adversely. Fraud detection encompasses a gamut of processes and tools aimed at identifying and averting unauthorized financial operations [1, 2]. It serves as a technological shield against a myriad of fraudulent activities, including but not limited to financial scams, telecommunication fraud, and online deceit. Such protective mechanisms find relevance across diverse sectors like finance, e-commerce, insurance, and telecommunications. Their primary objective? Timely detection and prevention of fraud to minimize financial damage and safeguard user interests.

The underpinnings of fraud detection largely rest on meticulous data analysis and modeling algorithms. These algorithms suss out irregularities and patterns suggestive of nefarious activities. Over the years, an array of techniques has been marshaled in the fight against fraud. For instance, real-time

DOI: 10.54254/2755-2721/48/20241181

monitoring and alert systems continuously track transactions and user behaviors. Any deviation from the norm, be it frequent personal information modifications or consecutive high-value transactions, can trigger immediate alerts. This prompts intervention by the relevant authorities for further scrutiny. Yet, as swindlers continuously refine their tactics, these systems confront new challenges daily. The escalating sophistication of these fraudulent schemes makes them harder to unearth through conventional means. Furthermore, vetting the legitimacy of a transaction is not instantaneous. The sheer volume of variables, patterns, and data points requires thorough analysis. This often means that the validation process lags, allowing some unscrupulous transactions to slip through the cracks.

Another stratagem in the anti-fraud toolkit is the use of rule engines. These are predicated on pre-defined criteria developed from industry standards or expert knowledge, acting as sentinels for potential malfeasance in real-time. However, the ever-evolving nature of fraud strategies challenges the efficacy of these rule engines. In certain traditional sectors, inter-organizational or inter-industry collaboration fortifies fraud detection. Sharing fraud intelligence insights enhances collective protective measures. But this collaborative approach struggles when dealing with decentralized, pseudo-anonymous constructs like the Bitcoin blockchain, complicating efforts to gauge the legitimacy of bitcoin transactions.

Enter data analysis and mining as potent tools in fraud detection. By dissecting vast swathes of historical data, algorithms can unearth hitherto unnoticed patterns indicative of fraud. Methods such as machine learning, statistical techniques, and artificial intelligence are instrumental in this endeavor. Herein lies their strength:

Comprehensiveness: Unlike traditional rule engines that operate within predefined parameters, data analysis can discern concealed patterns, offering a broader protective net.

Flexibility: As the Bitcoin market evolves, traditional methods may necessitate frequent rule updates. Data analysis techniques, however, can be fine-tuned to cater to new datasets and evolving needs, making them more adaptable.

Efficiency: When grappling with complex Bitcoin datasets, traditional methods might necessitate extensive rule formulation. In contrast, data analysis methods leverage algorithms and models for expedited processing.

Exploring an array of data analysis and mining techniques on the Bitcoin elliptic dataset, this study assesses the predictive accuracy of these methods against genuine transaction records. This analysis seeks to discern the most apt technique for the Bitcoin dataset, aiming to provide robust predictions regarding the legitimacy of Bitcoin transactions.

2. Fundamentals of Bitcoin and Elliptic Curve Cryptography

2.1. Introduction to Bitcoin

Bitcoin is a digital cryptocurrency that was invented by Satoshi Nakamoto in 2009. Unlike traditional currencies, Bitcoin does not require a central bank or other financial institution to issue and manage, but rather a decentralized transaction verification and bookkeeping system through blockchain technology.

The transaction process of Bitcoin is anonymous and decentralized. This makes Bitcoin an alternative currency in some countries and offers convenience, especially for those who wish to protect their personal identities.

Although Bitcoin has encountered some challenges along the way, it remains one of the most large-scale and widely used digital cryptocurrencies today, having a profound impact on global finance and economy, along with other cryptocurrencies and traditional financial systems.

2.2. Basics of Elliptic Curve Cryptography

ECC is an asymmetric encryption algorithm based on elliptic curve. The advantage of ECC over other encryption algorithms, for example, RSA [3], is though using shorter keys, it is able to achieve the same or higher security.

DOI: 10.54254/2755-2721/48/20241181

2.2.1. Principles of ECC

2.2.1.1. Elliptic curve. In general, elliptic curves can be represented by equations 1, where a, b, c, and d are the coefficients.

$$y^2 + aty + by = t^3 + ct^2 + dt + e \tag{1}$$

Binary cubic equation of plane curve is formula 2.

$$y^2 + aty + by = t^3 + ct^2 + dt + e \tag{2}$$

Substituting formula 2-2 into the second type of complete elliptic integral, formula 3 can be obtained.

$$f(x) = \int_c^x R|t\sqrt{t^3 + ct^2 + dt + e}|\,\mathrm{d}t \tag{3}$$

And because the formula for the circumference of the ellipse is shown in formula 4:

$$L = 4a\int_0^{\pi/2}\sqrt{1 - e^2 sin^2\theta}\,\mathrm{d}\theta = 4aE(e, \pi/2) \tag{4}$$

Early scientists called a curve like formula 1 an elliptic curve, and this concept is still in use today.

2.2.1.2. solving $E_p(a, b)$. Set that $E_p(a, b)$ Represents the set of points on an elliptic curve 5.

$$\{(x, y)|0 \le x \le p, 0 \le y \le p, both\ x\ and\ y\ are\ integer\} \sqcup 0 \tag{5}$$

The steps of solving $E_p(a, b)$ are as follows:

First for every integer x that satisfies $0 \le x \le p$, compute.

$$x^3 + ax + b(mod\ p) \tag{6}$$

Then detect if the result of 7 has a square root under module p, compute

$$y^2(mod\ p) \tag{7}$$

If there is a square root, It can be concluded that there is no corresponding point on the curve, otherwise work out the two square roots.

2.2.1.3. Addition rule. Define that $P, Q \in E_p(a, b)$, the following conclusions 8 and 9 can be drawn:

$$P + O = P \tag{8}$$

If $P = (x, y)$, then $(x, y) + (x, -y) = O$, To wit $(x, -y)$ is the Additive inverse of P.

Assume that $P = (x_1, y_1), Q = (x_2, y_2), P \ne Q$, then the result of $P + Q$ is determined by the following rules:

$$\begin{gathered} x_3 \equiv \lambda^2 - x_1 - \mathrm{x}_2(mod\ p) \\ y_3 \equiv \lambda(x_1 - x_3) - \mathrm{y}_1(mod\ p), \\ \text{where } \lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, P \ne Q \\ \frac{3{x_1}^2 + a}{2y_1}, P = Q \end{cases} \end{gathered} \tag{9}$$

2.2.1.4. Elliptic Curve Discrete Logarithm Problem (ECDLP). ECDLP is a mathematical problem widely used in elliptic curve cryptography. It is based on a variant of the discrete logarithm problem.

In elliptic curve cryptosystems, ECDLP means that at two points P and Q on a given elliptic curve, an integer k is found such that $Q = kP$. Here, P is a base point on an elliptic curve, and Q is the result of multiplying P by the integer k. In elliptic curve cryptography, P is referred to as the base point, k as the private key, and Q as the public key.

DOI: 10.54254/2755-2721/48/20241181

The difficulty of ECDLP is based on the property of the point addition operation on an elliptic curve, which forms an Abelian group on the elliptic curve. Because of the properties of associative, commutative and inverse elements of this operation, finding an integer k that satisfies $Q = kP$ becomes a difficult computational problem. Given k and P, it is easy to calculate Q according to the additive principle, but given P and Q, it is very difficult to find k.

The difficulty of ECDLP is one of the cornerstones of elliptic curve cryptography security. Many important cryptographic algorithms, such as elliptic curve digital Signature algorithm (ECDSA) and Elliptic curve key exchange algorithm (ECDH), depend on the difficulty of ECDLP problems. By choosing the right elliptic curve and key length for the right parameters, you can ensure that ECDLP problems cannot be resolved within an acceptable amount of time, thus protecting the security of the cryptosystem.

2.2.1.5. EL Gamal. ElGamal is an asymmetric encryption algorithm based on the discrete logarithm problem, proposed by Taher Elgamal in 1985. It is a public key cryptosystem that can be used to encrypt and decrypt messages and perform digital signatures.

ElGamal algorithm uses elliptic curves or large integer groups over finite fields to implement encryption and decryption operations.

ElGamal encryption includes the following steps:

The key generating process:

Select a prime number p and two random number g, x that are less than p, compute:

$$y = g^x \ mod \ p \tag{10}$$

(y, g, p) will be the public key, and x will be the private key.

The encryption process:

Assume that we have a plaintext message M, compute:

$$C_1 = g^k \ mod \ p, \ C_2 = y^k M \ mod \ p \tag{11}$$

The cyphertext $C = (C_1, \ C_2)$.

The decryption process:

$$M = \frac{C_2}{C_1^x} \ mod \ p \tag{12}$$

ElGamal algorithm has some advantages compared with other encryption algorithms, such as strong password security, simple key distribution and so on. However, it also has some disadvantages, such as the length of the encrypted ciphertext is long, and the encryption and decryption operation is relatively slow.

2.2.2. The use of ECC in bitcoin

In Bitcoin, ECC (Elliptic Curve Cryptography) is widely used for address generation and digital signatures. Using the SECP256K1 elliptic curve parameter, ECC provides a secure, efficient and compact encryption solution.

In terms of address generation, Bitcoin uses ECDSA (ECC based digital signature algorithm) to generate user wallet addresses. Multiplication of the private key and elliptic curve points produces the public key and obtains the address through the hash function.

In terms of digital signatures, transactions need to be digitally signed to verify validity. The sender uses the private key to sign the transaction, and the receiver uses the corresponding public key for verification.

The application of ECC in Bitcoin brings the following advantages: high security, high computational efficiency, and low storage space.

Overall, ECC provides a reliable cryptographic foundation for address generation and digital signatures in Bitcoin, while also improving the security and efficiency of the system.

DOI: 10.54254/2755-2721/48/20241181

3. Principles of Proposed Models

3.1. Graph Convolutional Networks

Deep learning on graph structured data has already become a subject attracting increasing interest [4-8]. Given the inherent combinatorial complexity of graph structures, practical applications do face scalability challenges. Since the number of nodes and edges in a graph can be very large, processing and analyzing these large-scale graph data requires efficient algorithms and systems. However, significant progress has been made in addressing these challenges, for example Graph database [9], Distributed graph computing framework [10], Graph embedding [11], graph neural network [12-14], etc. These advances provide a feasible solution for dealing with the inherent combinatorial complexity of graph structures, and remarkable results have been achieved in practical applications.

Specifically, we consider GCNs. Based on the idea of Convolutional Neural Networks, it extends convolution operations to graph structured data, and can effectively perform tasks such as node classification and link prediction. A GCN model consists of multiple graph convolution layers. GCN uses well-designed techniques to effectively extract features from graph data for a variety of applications such as node classification, graph classification, link prediction, and more. In addition, GCN is capable of generating graph embeddings, facilitating further analysis and application. The kernel part of GCN are as follows:

Given a batch of graph data containing N nodes, each node possesses its own features. Assuming the features of these nodes shape an N×D matrix X, the relationships between each node subsequently form an N×N dimensional matrix A, commonly referred to as the adjacency matrix. X and A are the inputs to our model. The way of its propagation between layers can be described by formula 13, where $\tilde{A}$ represents $A + I$, $\tilde{D}$ represents the degree matrix of $\tilde{A}$, shown by formula 14. H represents the feature of each layer, and H of the output layer is W. σ is the activation function.

$$H^{(l+1)} = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}) \tag{13}$$

$$\tilde{D} = diag(\sum_j \tilde{A}_{ij}) \tag{14}$$

The initial embedding matrix comes from the node features, i.e., $H^{(0)} = X$. Assume that there are L graph convolution layers. The output layer is typically a softmax layer, where $H(L)$ represents the final hidden features obtained after L layers of graph convolutions.

3.2. Graph Attention Networks

GAT is an improved Graph neural Network model based on GCN. Some of the main improvements of GAT over GCN are discussed below.

3.2.1. Graph attention mechanism

The key of the graph attention network is the graph attention layer, the processing object of the graph attention layer is each node of the whole graph. Assume that there are N nodes, and the feature dimension of each node is F, then the output obtained through the graph attention layer is $N \times F'$,where F' represents the feature dimension of the output layer. Use $\mathrm{h} = \{h_1, h_2, \ldots h_N\}$ to represent the input data, $\mathrm{h}' = \{h_1', h_2', \ldots h_N'\}$ to represent the output data, where h_i' is a F' -dimension vector. An Attention coefficient e_{ij} is also introduced to describe the influence of node j on node i. In order to obtain h_i' , we need to consider the attention coefficient for node i of all nodes adjacent to node i. Thus formula 15 was introduced.

$$\vec{h'}_i = \sigma(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathrm{W} \vec{h}_j) \tag{15}$$

DOI: 10.54254/2755-2721/48/20241181

$\mathcal{N}_i$ represents The field of all nodes adjacent to node i, $\vec{h}_j$ represents the eigenvector of node j, and W is a weight matrix of size $F' \times F$. α_{ij} can be computed by formula 16. And by bringing the calculation formula of e_{ij} into formula 3-4, α_{ij} can be calculated by formula 17.

$$a_{ij} = softmax_j(e_{ij}) = \frac{(e_{ij})}{\sum_{k \in \mathcal{N}_i} exp(e_{ik})} \tag{16}$$

$$a_{ij} = \frac{exp(LeakyReLU(\vec{a}^T|\mathbf{W}\vec{h}_i||\mathbf{W}\vec{h}_j|))}{\sum_{k \in \mathcal{N}_i} exp(LeakyReLU(\vec{a}^T|\mathbf{W}\vec{h}_i||\mathbf{W}\vec{h}_k|))} \tag{17}$$

3.2.2. Multi-head attention mechanism

In order to better capture complex relationships between nodes, GAT employs a multi-head attention mechanism that can learn multiple sets of different attention weights in parallel. Each head has its own attention mechanism and weight parameters, allowing it to extract different model features. Finally, the final node representation is obtained by concatenating or averaging the results of multiple heads.

3.3. Voting Mechanisms

Voting is an ensemble learning technique that can be used to make predictions in classification models or regression models. In the voting method, the prediction results of multiple basic learners are combined by voting to reduce variance and improve the overall prediction performance. During the voting process, each base learner is considered a voter, each category or value is considered a contender, and a winner is selected based on the voting results.

3.3.1. Hard voting

Hard voting assumes that the category with the most votes is the winner.

3.3.2. Soft voting

Soft voting differs from hard voting in that it takes the confidence of the predicted outcome into account. Each base learner calculates a confidence score for each category or value, which is then weighted and averaged to get the final prediction of the soft vote. This approach allows even if a basic learner occasionally makes a wrong prediction, as long as its confidence is high, its prediction results will still have a positive impact on the overall prediction results. By considering confidence, soft voting can improve accuracy and robustness, and has some fault tolerance in the face of noise interference.

4. Experiments

4.1. Data Source, Cleaning, and Pre-processing

4.1.1. Data source

The author uses the Elliptic dataset as data source. The Elliptic dataset record real Bitcoin transactions that have been judged licit or illicit. A graph whose nodes represent Bitcoin account addresses and edges represent transaction relationship is constructed to record bitcoin transactions. If the entity (or account address) initiating the transaction is deemed licit, then the transaction will be identified to belong to licit category, and vice versa. The Elliptic dataset totally covers 203,769 node transaction records over a period of six months. 21% of the transactions are labeled licit, and 2% are labeled illicit.

4.1.2. Data cleaning and pre-processing

First convert the categories to numeric labels, the author recorded the licit transactions as class 1, illicit as class 2 and the rest transactions that are not labeled as class 0. After the prior processes, the author combined features and categories and sorts them by node ID, then created an index of the edge, loads

DOI: 10.54254/2755-2721/48/20241181

the node features and converts them to tensors. Finally, the author split the dataset into training set and test set, with 15% of the transaction data as test set with train_ test_ split function in sklearn.

4.2. Model Building and Prediction

4.2.1. Model building

The author totally built 5 models, 2 for GCN (a 2-layer GCN and a 4-layer GCN), 1 for GAT and 2 for GAT with voting mechanisms (one with hard voting and the other with soft voting).

4.2.1.1. GCN. The researcher trained GCN models utilizing the Adam optimizer for a total of 1,000 epochs, eventually settling on a learning rate of 0.001. Two distinct GCN models were experimented with: one with 2 layers and another with 4 layers.

Given the priority of detecting illicit transactions in fraud detection tasks, and considering that these illicit transactions represent only a small fraction of the entire dataset, a weighted cross-entropy loss was employed. This approach was chosen to amplify the impact of the illicit samples during the training phase. After careful hyperparameter tuning, appropriate weights for illicit, licit, and unknown transactions were determined. For the 2-layer GCN model, the weight dimensions of the convolutional layers were set to 165x360 and 360x1. Conversely, the 4-layer GCN model had its weight dimensions configured as 165x360, 360x512, 512x256, and 256x1 for each layer respectively.

4.2.1.2. GAT. The learning rate and number of training epochs of GAT were set the same as those of the two GCN models. The settings of proportion between illicit transaction, licit transaction and unknown was the same as the proportion in the two GCN models mentioned above.

4.2.1.3. GAT with hard voting. The prediction results of models with prediction probability greater than or equal to 0.8 are selected as the alternative set to participate in voting. If there is no model with prediction probability greater than or equal to 0.8, the top 10 models with the largest prediction probability are selected to participate in voting.

For each sample, the number of occurrences of each category in the alternative set is counted, and the category with the most occurrences is selected as the final prediction result. If there are multiple categories that appear most frequently, randomly select one of them as the final prediction result. Finally, when calculating the accuracy of the model on the test set, the actual label is compared with the prediction results obtained by voting, the number of samples predicted correctly for each category is counted, and the accuracy is calculated.

The average of the prediction results of all the models participating in the voting is used as the final prediction result.

4.2.1.4. GAT with soft voting. The average of the prediction results of all the models participating in the voting is used as the final prediction result. Other mechanisms are the same as hard voting.

4.2.2. Prediction Result

Table 1 shows the prediction result of the models the author built, including the accuracy of predicting licit transactions, illicit transactions and the total accuracy.

Table 1. Prediction result.

Model	Licit accuracy (%)	Illicit accuracy (%)	Overall accuracy (%)
2-layer GCN	79.22	75.66	78.85
4-layer GCN	78.01	75.81	77.78
GAT	89.54	92.23	89.79
GAT with hard voting	91.08	87.39	90.71
GAT with soft voting	90.61	88.12	90.35

DOI: 10.54254/2755-2721/48/20241181

It can be inferred from the table that in terms of overall accuracy, models using voting method perform better than that without voting, while hard voting seems to be more efficient than soft voting. GAT performs better than GCN, and the 2-layer GCN performs better than 4-layer GCN. When it comes to illicit accuracy that we pay more attention to in the process of fraud detection, models using GAT still performed better than GCN models overall, but it is worth noting that GAT without voting mechanism unexpectedly performed better than the two GAT models with voting mechanism.

The author attempted to analyze the reason why these results occur. The first point the author tried to find out was the reason why the 4-layer GCN worked out a result that is almost no difference in illicit accuracy compared to the 2-layer GCN, and it even has poorer performance when it comes to total accuracy. The author used matplotlib to show the whole training process, and the reason was found by observing the accuracy during training. After training for about 40 epochs, the 4-layer GCN faced the problem of overfitting. By contrast, the 2-layer GCN did not face this problem and performs basically equal or even better than the 4-layer GCN. It is widely known that simple models tend to be more generalizing and better able to adapt to unknown data. Simple models typically have fewer parameters and simpler structures, and are more limited to learning overall patterns rather than details in the training data.

The author also tried to figure out the factors that affect the efficiency of voting mechanism. One of the reasons that might affect the efficiency might be as the underlying GAT model already has high accuracy and captures the characteristics of the data well, the effect of including voting may be limited. The deciding factor that the author indicates might be the diversity among models. To improve accuracy through voting, different models need to have diversity. They should have different error tendencies in their predictions so that integration can compensate for each other's shortcomings.

5. Conclusion

While traditional statistical methods offer a starting point for anomaly detection in Elliptic datasets, they present various limitations when navigating the complex world of financial transaction networks. Specifically, they may struggle to capture subtle anomaly patterns due to the intricate non-linear relationships inherent in such data. When it comes to high-dimensional data, these methods can suffer from the curse of dimensionality, leading to overfitting or incorrect parameter estimates. Other challenges include data distribution uncertainties and outliers that depart from the norm. Given these constraints, a comprehensive approach that combines machine learning and deep learning techniques appears more promising for effective anomaly detection within Elliptic datasets. The researchers delved into the creation of diverse models, assessing their relative performances. Notably, the GAT models outperformed the GCN models, making them a preferred choice for predictions on the Elliptic dataset. Although dual models with a voting mechanism did not significantly surpass the standalone GAT in the study, the inherent potential of voting methods suggests possible improvements through model variation. In the end, those employing these models must make a strategic choice: pursue the potential increased accuracy of a model with a voting mechanism or avoid the extended training durations that such mechanisms typically demand.

References

[1] Weber, M., Domeniconi, G., Chen, J., Weidele, D. K. I., Bellei, C., Robinson, T., & Leiserson, C. E. (2019). Antimoney laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics.

[2] Nakamoto, S. (2008). Bitcoin: a peer-to-peer electronic cash system.

[3] Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2), 120-126.

[4] Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2014). Spectral Networks and Locally Connected Networks on Graphs. In ICLR.

[5] Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In NIPS.

DOI: 10.54254/2755-2721/48/20241181

[6] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural Message Passing for Quantum Chemistry. In ICML.
[7] Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. In NIPS.
[8] Li, Y., Tarlow, D., Brockschmidt, M., & Zemel, R. (2016). Gated Graph Sequence Neural Networks. In ICLR.
[9] Fu, Z., Wu, Z., Li, H., Li, Y., & Hu, X. (2017). GeaBase: A High-Performance Distributed Graph Database for Industry-Scale Applications. 2017 Fifth International Conference on Advanced Cloud and Big Data (CBD). IEEE.
[10] Tang, Z., He, M., Fu, Z., & Yang, L. (2020). Incgraph: an improved distributed incremental graph computing model and framework based on spark graphx. IEEE Transactions on Knowledge and Data Engineering, PP(99), 1-1.
[11] Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: online learning of social representations. ACM.
[12] Chen, J., Ma, T., & Xiao, C. (2018). FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In ICLR.
[13] Kipf, T. N., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. In ICLR.
[14] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018). Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In KDD.