Deep Graph Kernels for Inferring Bitcoin Transactions Dynamics [*]

Pankaj Kumar[†]

[*]Interim thesis report

[†]Skolkovo Institute of Science and Technology, Skolkovo Innovation Center, Building 3, Moscow 143026 Russia. Email: pankaj.kumar@skolkovotech.ru

1 Introduction

Bitcoin is a decentralized peer to peer electronic payment system in which transactions are performed with no central authority or banks to authorize it. The Bitcoin transactions management and its issuance is carried out collectively by the network. The first Bitcoin specification and proof of concept was published by Satoshi Nakamoto (Nakamoto, 2009) in 2009 cryptographic mailing list. Since then, the community has grown exponentially with many developers working on Bitcoin.

Bitcoin payments use public keys encryption, where, payers and payees are identified by hashed public keys of their Bitcoin wallets. The public keys are generated by ECDSA (Elliptic Curve Digital Signature Algorithm), based on calculations of elliptical curves over finite space. Suppose, Alice sends "x" coins to Bob, then an unencrypted transaction attaching Bob's public key is broadcast over the Bitcoin network using her private key. The signature on transaction verifies all users for its authenticity (current owner of coin) by looking at complete history of transactions.

1.1 Blockchain

The complete record of transactions is in a coded form in a data structure called *blockchain*, which is a sequence of records called *blocks*. Each block contains a group of transactions that have been sent since the previous block, with integrity check all the way back to the first one, the genesis block.

Any users enter the Bitcoin system by trading non-digital currencies at Bitcoin market exchange, or by mining coins, which involves solving a cryptographically hard problem for which he/she is suitably rewarded a fixed number of Bitcoins and transactions is validated on the network. The working of mining, core to *blockchain*, is mathematically explained better by Göbel et al. (2015).

At first, miner M computes a block hash h over a unique ordering of the hashes of all the transactions that it is intending to incorporate into its next block B. Then, input of the block solution s_{i-1} is taken at the head of its current version of the *blockchain*. If we concatenate the strings by the symbol $+$, the cryptographic problem that M has to solve is: compute a SHA-256 hash

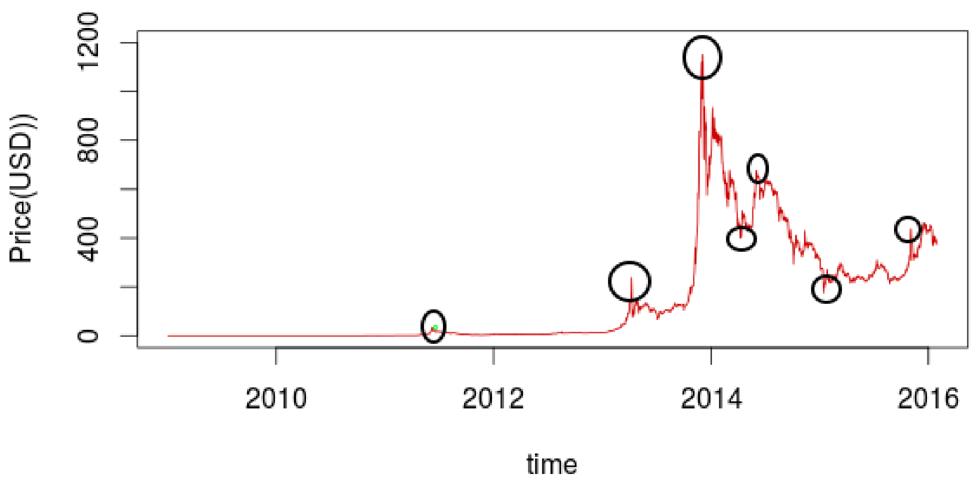
$$s_i = (n + h + s_{i-1}), \tag{1.1}$$

such that s_i has at least a specified number x of leading zeros where $x \sim 64$.

1.2 Market & Bitcoins

Bitcoin started trading on Mt.Gox (its market share was over 80% on the BTC/USD spot market), the largest online Bitcoin exchange at 5 cents. Over the next two years, BTC/USD exchange rose slowly, but in early April 2013, the price per BTC dropped from $266 to around $50 and then rose to around $100. This major bubble was succeeded by another one on 29 Nov, 2013, when, the price of a bitcoin reached an all-time high of US$1224.76 on 29 November 2013, then dropped into the $200-$300 range. Metaphorically, the price rise/drop is expressed as skippyball. Once skippyball pitched high in the air, the first bounce is big, then lower, and lower, until it is flat on the ground, ready for a new throw upward. Taking market price data from Blockchain, the most popular Bitcoin wallet [1], the bubbles can be visualized in the time series plot 1.1 with circles representing bubbles.

Figure 1.1 Market Price (USD)



According to an economic explanation (Lo and Wang, 2014; Kristoufek, 2015) for Bitcoin bubble in April, 2013, the financial crisis in Cyprus, triggered large numbers of people to converted their euros into digital Bitcoins. The real cause of the Bitcoin bubble and burst is still unexplored research territory , but it encourages researchers to model the the underlying

[1]Market Price in USD https://blockchain.info/charts/market-price

system of interacting agents to gain a better understanding of the underlying phenomena. The first step in this direction would be to get transaction network, identify its structural parameters, like walk or path and find its correlation with market price. Then, comes the understating of the network behaviour to have concrete explanations to support any theory.

Using complete list of transactions, parsed from blockchain, Kondor et al. (2014) reconstructed transaction network and applied unsupervised identification to get the most important features of the time variation of the network. By applying Principal Component Analysis to the matrix constructed from snapshots of the network at different times, they showed that structural changes in the network accompany significant changes in the exchange price of Bitcoins. The one fall-back about this paper is scalability, which makes it unsuitable for application having nodes greater than 10000 nodes (Zager and Verghes, 2008).

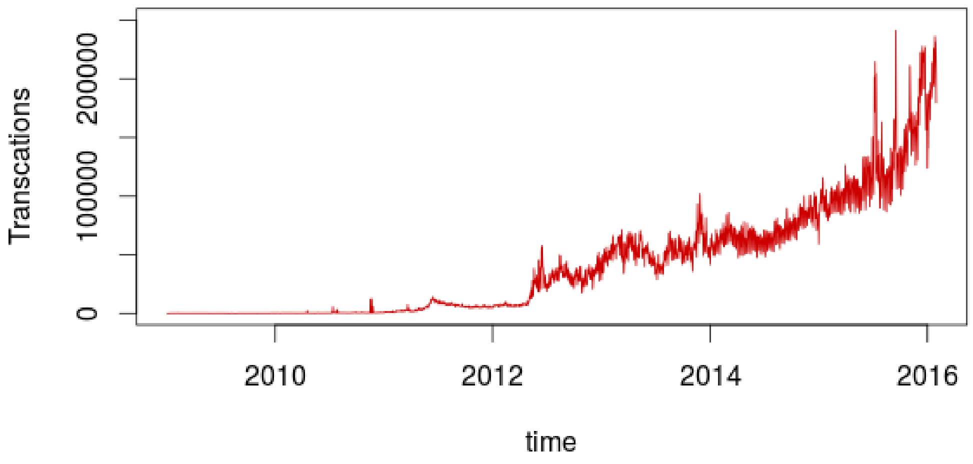
1.3 Graph Similarity & Subgraph Matching

The above solution proposed by Kondor et al. (2014) falls on the category of basic graph similarity and subgraph matching algorithms, which is not scalable with increasing number and the size of graphs (eg: bitcoin 1.2). There are challenges on proposing meaningful metrics to capture different notions of structure, designing algorithms that can calculate these metrics, and finally finding approximations or heuristics to scale with graph size if the original algorithms are too slow.

Most of graph isomorphism algorithms and its generalizations are exponential and, thus, not applicable to the large graphs that are of interest today. The other methods for graph similarity and graph sub-matching, like features extractions, iterative methods and tensors analysis, including some come with their drawbacks (Zager and Verghes, 2008).

In the recent paper on graph kernels, Yanardag and Vishwanathan (2015) used ”Deep Graph Kernels”, a unified framework to learn latent representations of sub-structures for graphs to compare graphs. As Bitcoin transaction network graph is increasing everyday, It would be interesting to capture its structural dynamics by method proposed in the paper.

Figure 1.2 Daily Transactions



1.4 Motivation

The volatile rise and fall of Bitcoin, cryptological mining, complex distributed database, blockchain and evolving network dynamics promoted me to efficiently measures the structural change (eg. walks) of a dynamic large-scale graph as well as the similarity between two graphs with reference to price change.

In simple words, the thesis attempts to address two important questions:

1. How much is a daily bitcoin transaction graph transformed over time or by a significant event (bubbles)?
2. How structurally similar are two daily bitcoin transactions graphs ?

1.5 Objective

The objective of thesis are following:

1. Given complex cryptic distributed blockchain, parsing it to extract transaction attributes in the form. complete

$Transaction_{From}$	$Transaction_{To}$	$Value$	$Timestamp$

2. Get daily transactions graphs. complete
3. Given two graphs $G_1(n_1, e_1)$ and $G_2(n_2, e_2)$, find an algorithm to calculate the similarity index between the two graphs. complete.
4. Given a graph time series, where there are T number of graphs, find approximate subgraphs that occur in a subset of the T graphs. partial complete

2 Data

The blockchain is a transaction database of the Bitcoin. Once Bitcoin core is setup at local machine, blockchain is automatically downloaded. Every full node participating in the Bitcoin network has the same copy. As of now there is more than 60 GB of Bitcoin blockchain dataset, which makes it difficult to parse the raw blockchain data. Most of the previous studies (Ron and Shamir, 2013) employed a forked version of bitcointools [2], but from the bitcoin clients 0.8 version, it indexed the full blockchain using LevelDB instead making the publicly available bitcointools obsolete. Other well know open source blockchain parser like blockparser [3], BitIodine (Spagnuolo, 2013) etc are almost undocumented projects, where some appear to not even work. A paper (Fleder et al., 2015) led me to BitcoinArmory project, which requires a dozen of manual interventions to get installed, but still doesnt work.

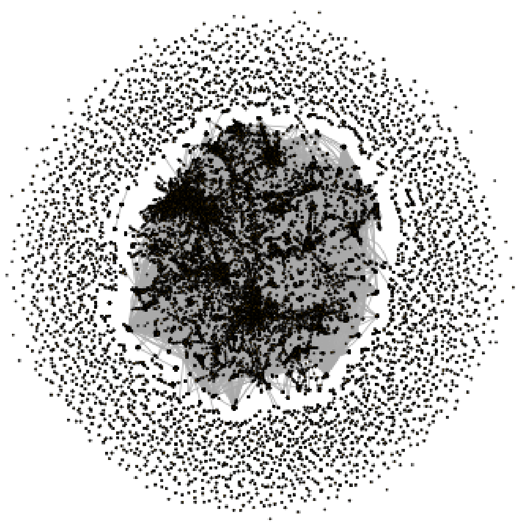
We then started to look for ready-made SQL database, which pointed to BitcoinABe. This python based library reads the Bitcoin block file, transforms and loads the data into a SQL database. But, Its takes more than two weeks to dump the data, which was not feasible option. We downloaded postgres database dumps of the bitcoin-ruby-blockchain database generated by webbtc [4]. Then, with slight modification to open repository BitIodine (Spagnuolo, 2013) code, we parsed through the blockchain, and wrote wrapper classes that extracted the relevant information required to construct the transaction graph. The market price data was scarped from website `https://blockchain.info/` to plot graphs in R.

[2]`https://github.com/gavinandresen/bitcointools`
[3]https://github.com/znort987/blockparser
[4]`http://dumps.webbtc.com/bitcoin/`

3 Bitcoin Transaction Netowrk

With the overall transactions records parsed from the blockchain in human readable form, we construct a a weighted directed transaction graph that gives an intuition towards the flow of Bitcoins from one key to the other,the directed edge represents a particular transaction from a source address to a target address and weight represents the value of the transactions. For our experiments, we constructed graphs for 45 days between and after 10 April, which was first major bubble in Bitcoin history. we wanted to first confirm the correlation between market price and network dynamics, before proceeding to analyse whole data.

Figure 3.1 Daily transaction graph for a typical day (April 8, 2013)



4 Graph Isomorphism

Long in the purview of researchers, graph isomorphism is a well-studied problem in literature. There are several approaches proposed to solve variations of the problem. Though graph isomorphism is a classic open problem in the algorithmic theory of random graphs, but recent work by Babai (2015) showed that it can be solved in in quasipolynomial ($\exp\left((\log n)^{O(1)}\right)$) time. Where, the best previous bound for graph isomorphism was $\exp(O(\sqrt{n \log n}))$; n is the number of vertices.

Another approach of attacking graph isomorphism is based on features extraction. The λ-distance, a spectral method and levenshtein distance which defines the distance between two graphs as the distance between their spectra (eigenvalues) has been studied thoroughly (Koutra et al., 2015). The study by Li et al. (2012) proposes an SVM-based approach on extracted features (average degree, eccentricity, number of nodes and edges, eigenvalues, clustering coefficient, diameter etc.) to perform graph isomorphism. Other techniques includes, computing edge curvatures under heat kernel embedding, comparing graph signatures consisting of summarized local structural features and a distance based on graphlet correlation (Koutra et al., 2015). Though these methods are powerful and scale well, but depending on the statistics that are chosen, it is possible to get results that are not intuitive. For example, it is possible to get high similarity between two graphs with very different node set size, which is not always desirable. Adding to the same, feature vector transformation leads to loss of topological information.

Next approach is based on iterative methods, which is based on the concept to find the "best" correspondence between the nodes of the two graphs. The research direction attempting to solve the graph alignment problem is flooded with the methods span from genetic algorithms to decision trees, clustering, expectation-maximization, similarity flooding algorithm (Melnik et al., 2002), message-passing algorithm for aligning sparse networks and belief propagation (Koutra et al., 2015), to name few of them. The advantages of all enlisted methods are their speed and simplicity, but they do not take into account information about the graph structure. On graph similarly measure with given node correspondence, Koutra et al. (2015) proposed DELTACON , a principled, intuitive, and scalable algorithm that assesses the similarity between two graphs on the same nodes using Fast Belief Propagation on real graphs from ENRON e-mail exchange and brain scans. The specific application to node correspondence make its unsuitable for dynamic graphs.

Going in the research direction of tracking changes in networks over time, spotting anomalies and detecting events, Lee et al. (2015) graph similarity approach is based on random walk with restart (RWR) algorithm with intergraph compression to transform representation of graph. The method is efficient in space requirement and produces results more quickly and accurately over conventional graph transformation schemes. The methods fails in picking Euclidean distance as measurement of matrix distance, as it become weakly discriminant when we have multidimensional and sparse data. Another work by Mheich et al. (2015) on measure graph similarity takes account for vertices, edges and spatiality at the same time, which is unseen in literature.

Another important approach which work directly on the graphs without doing feature extraction is called graph kernels. It characterize graph features in a high dimensional space and thus better preserve graph structures. Most of graph kernels are instances of the family of R-convolution kernels proposed by Haussler (1999), which define graph kernels by comparing all pairs of isomorphic substructures under decomposition, and a new decomposition will result in a new graph kernel. It can be categorised into classes based on comparing all pairs of walks, paths, cycles (Aziz et al., 2013), trees and graphlets in polynomial time (Vishwanathan et al., 2010).

One of the popular graph kernel, random walk graph kernel (Gärtner et al., 2003; Kashima et al., 2003) counts matching walks in two input graphs. It is prohibitively expensive, requiring $O(n_6)$ runtime and suffers from the problem of tottering and halting. The computation of the above was greatly improved by Vishwanathan et al. (2007) using iterative methods, including those based on Sylvester equations, conjugate gradients, and fixed-point iterations. Other improvements by Kang et al. (2012) takes account for just unlabeled graphs with the normalized weight matrix. Tottering problem was solved by Mahé et al. (2004) by doing special transformation on the input graphs, but it suffer from adverse computation time ($O(n)$ to $O(n_2)$ and does not show a uniform improvement of classification accuracy. Replacing walk by path Borgwardt and Kriegel (2005), solved the tottering and halting problem, but dense matrix representation for connected graphs, may lead to memory problems on large graphs. Adding to the same, the recent paper by Sugiyama and Borgwardt (2015) claimed geometric random walk kernels suffers from the problem referred to as halting: Longer walks are downweighted so much that the similarity score is completely dominated by the comparison of walks of length 1.

Other graphs kernels too suffer from disadvantages. For example, Optimal Assignment Kernels and Edit-Distance Kernel are not positive definite in general; Subtree Kernel runtime grows exponentially with the recursion depth of the subtree like patterns; Cyclic Pattern Kernel restrict their attention to scenarios where the number of simple cycles in a graph dataset is bounded by a constant and Graphlet Kernel's common solutions not feasible on labeled graphs (Vishwanathan et al., 2010).

Recent work on graph kernels tries to capture structural information encoded in node labels, attributes, and edge information to come up "propagation kernels", which can be used to construct kernels for many graph types, including labeled, partially labeled, unlabeled, directed, and attributed graphs (Neumann et al., 2015). Propagation kernels were originally defined and applied for graphs with discrete node labels (Shervashidze et al., 2011). In another work by Orsini et al. (2015), the author upgrade generalization of Weisfeiler-Lehman (Shervashidze et al., 2011) and propagation kernels (Neumann et al., 2015) to continuous attributes by using graph and vertex invariants.

In the recent KDD 2015 paper by Yanardag and Vishwanathan (2015), the author used language modeling and deep learning to learn latent representations of sub-structures for graphs. Their framework leverages the dependency information between sub-structures by learning their latent representations. Using three popular graph kernels, namely Graphlet kernels, Weisfeiler-Lehman subtree kernels, and Shortest-Path graph kernels, their models proves to be improves the classification accuracy, robust to random noise and computationally efficient.

5 Network & Price Correlation

The computationally efficient kernel methods requires kernel functions, an inner product of two vectors in a future space for all computations. In order to apply kernel methods to graph classification, we first need to define a kernel function between the graphs. Graph Kernels aim at computing similarity scores between graphs in a dataset, performed by using pattern derived out of frequent subgraph mining. Patterns are used as features for graph comparison (Deshpande et al., 2005). Number of kernels taking account of positive semidefinite is proposed for structure data in various works, good summary of the same is enlisted in Vishwanathan et al. (2010). The basic idea is based on decomposing an object into substructures and a feature vector, composed of the counts of the substructures.

In this thesis due to structure of our transaction graph, we consider labeled directed graphs without parallel edges. To measure similarity between two graphs, we use Gärtner et al. (2003) approach to describe a labeled graph as a feature vector is to count label paths appearing in the graph. A label path is produced by random walks on graphs, and then, kernel is defined as the inner product of the count vectors averaged over all possible label paths, which is regarded as a special case of marginalized kernel (Vishwanathan et al., 2010).

Taking two graph G_1, G_2, let $E_\times$ denote the adjacency matrix of their direct product $E_\times = E(G1 \times G2)$, and $V_\times$ denote the vertex set of the direct product $V_\times = V(G1 \times G2)$. With a sequence of weights $\lambda = \lambda_0, \lambda_1, \ldots$ ($\lambda_i \in R$; $\lambda_i \geqslant 0$ for all $i \geqslant N$) the direct product kernel is defined as:

$$K_\times(G_1, G_2) = \sum_{i,j=1}^{|V_\times|} \left[\sum_{k=0}^{\infty} \lambda_k E_\times^k \right]_{ij} \tag{5.1}$$

The graph kernel $K(G_1, G_2)$ is computed efficiently computing matrix power series $\lim_{n\to\infty} \sum_{i=0}^{n} \lambda_i E^i$, which is explained in Gärtner et al. (2003) by taking exponential and geometric series. The graph kernel is normalised to get similarity index in our case. In simple words, elegant computation involves, computing walk of length k by looking at the k^{th} power of the adjacency matrix, then constructing direct product of the graph G_1, G_2 and counting walk on the product graph $K_\times(G_1, G_2)$. $E^k(i,j) = c$ means that c walks of length k exist between vertex i and vertex j.

The above method is implemented in R package Rchemcpp (Klambauer et al., 2015), we used it for our graph to calculate similarity index. We then plot graphs to see any correlations between graphs and market close price. The same experiments were performed using logarithms of the closing price. Pearson correlation coefficients between both the prices are more than 0.70. Experiments with other similarity measure were also performed to checked robustness of the method used.

Visual inspection shows a good correspondence between price and similarity index, which is in line with the high high correlation coefficients (0.77) between two datasets.

We plan to use other methods discussed in Vishwanathan et al. (2010) to check for expensive runtime $O(n^6)$, tottering and halting (Borgwardt and Kriegel, 2005) problem on above discussed method.

Figure 5.1 Daily transactions price correlation with network structure.

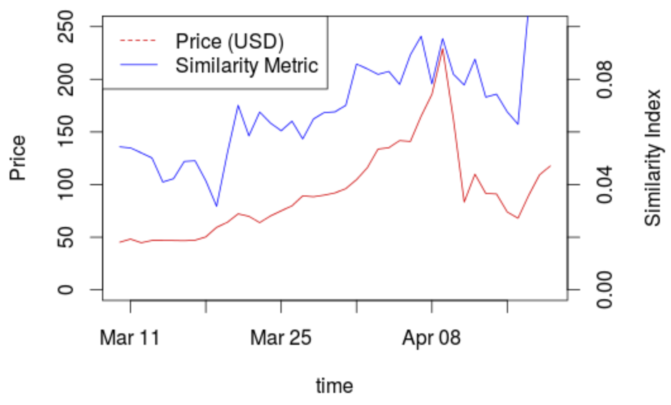
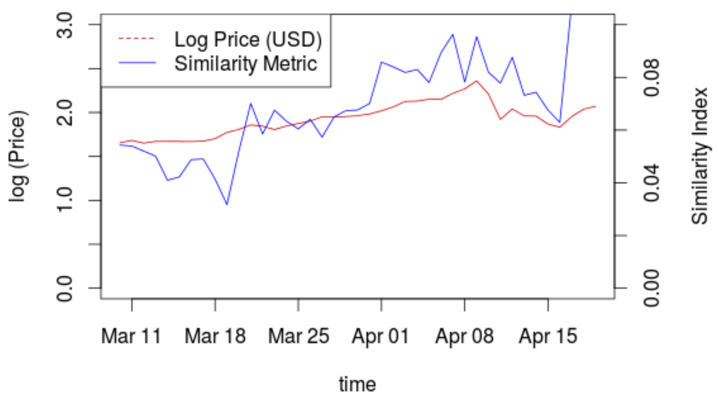


Figure 5.2 Daily transactions price (log) correlation with network structure.



6 Deep Graph Kernel

Deep Graph Kernels (Yanardag and Vishwanathan, 2015) is not so deep learning frame work to learn latent representations of sub-structures for graphs, which are not independent. The framework leverage on the dependency information between sub-structures by learning their latent representations.

Graph Kernels are produced using R-convolution, where a graph is to recursively decomposed into into atomic sub-structures to define a kernels between them. Mathematically, It can be expressed as:

$$K(G_1, G_2) = \left\langle \Phi(G_1), \Phi(G_1) \right\rangle_{\mathcal{H}} \tag{6.1}$$

Where, $\Phi(G)$ is denote a vector which contains counts of atomic sub-structures. The representation in the equation 6.2 suffers from two basic problems. First, the sub-structure are not independent. Second, the dimension of the feature space often grows exponentially, as substructure grows, which leads to "diagonal dominance", that is, a given graph is similar to itself but not to any other graph in the dataset. But, we need a kernel matrix where all entries belonging to a class are similar to each other, and dissimilar to everything else. To alleviate this problem, we define alternative graph kernel as:

$$K(G_1, G_2) = \Phi(G_1)^T \mathcal{M} \Phi(G_1) \tag{6.2}$$

$\mathrm{L}_{SG} = \sum_{i=1}^{|\mathcal{D}|} \sum_{i-K \leqslant k \leqslant i+K, k \neq i} -\log p(c_k|w_i)$
$= \sum_{i=1}^{|\mathcal{D}|} \sum_{i-K \leqslant k \leqslant i+K, k \neq i} -\log \frac{e^{\mathbf{c}_k^T \mathbf{w}_i}}{\sum_{v=1}^{V_c} e^{\mathbf{c}_v^T \mathbf{w}_i}}$
$= \sum_{i=1}^{|\mathcal{D}|} \sum_{i-K \leqslant k \leqslant i+K, k \neq i} -\mathbf{c}_k^T \mathbf{w}_i + \log \sum_{v=1}^{V_c} e^{\mathbf{c}_v^T \mathbf{w}_i}$
(6.3)

where $\mathcal{M}$ represents a $|\mathcal{N}| \times |\mathcal{N}|$ positive semi-definite matrix that encodes the relationship between sub-structures and $\mathcal{N}$ represents the vocabulary of sub-structures obtained from the training data. Now, the important task is design $\mathcal{M}$, which can be computed by learning latent representations of sub-structures or by edit-distance relationship between sub-structures.

The language modeling and deep learning techniques was used to compute $\mathcal{M}$ with intuition that different sub-structures compose graphs in a similar way that different words form sentences when used together. The list of decomposed sub-structures for each graph is then treated as a sentence that is generated from a vocabulary $\mathcal{V}$ where vocabulary $\mathcal{V}$ simply corresponds to the unique set of observed sub-structures in the training data. The linear co-occurrence relationship of sub-structure is created using modified random sampling scheme, taking account of neighborhoods. That is, whenever we randomly sample a graphlet G, we also sample its immediate neighbors. Then is corpus is generated according to particular graph kernels, for example, for shortest graph kernels, whenever shortestpath sub-structure is generated, all possible shortest-path sub-structures are also collected that share the same source node, and treat them as co-occurred.

Once corpus is generated, the model is build by using CBOW or Skip-gram algorithms and trained with Hierarchical softmax or negative sampling. If Let s represent an arbitrary sub-structure from a vocabulary $\mathcal{V}$, and ϕ_s represent learned vector representation of s. Matrix $\mathcal{M}$ is calculated such that, each entry on the diagonal is $\mathcal{M}_{ii}$ computed as $\langle \phi_i, \phi_i \rangle$ where ϕ_i corresponds to learned d-dimensional hidden of sub-sequence i and $\mathcal{M}_{ii}$ where i =j and $1 \leqslant i \leqslant |\mathcal{V}|$ (resp. j). After computing the $\mathcal{M}$ matrix, it is plugged in Equation 6.2 to get deep graph kernels.

The conducted experiments with diverse real life data, confirms that deep kernels improves the classification accuracy, robust to random noise and computationally efficient.

Most of the graph kernels (Vishwanathan et al., 2010) are designed for unlabeled graphs or graphs with a complete set of discrete node labels , but graphs with continuous node attributes is catching up fast (Feragen et al., 2013). But these graph kernels can only handle graphs with complete label or attribute information in a principled manner. These may be efficient, but for specific graph and if flexible, computation is memory and/or time consuming. The propagation kernels by Neumann et al. (2015) overcomes all the above problems. We exploit the propagation kernels in the deep framework to have similarity index between two graph. The propagation kernels measure the similarity between two graphs by comparing node label or attribute distributions after each step of an appropriate random walk.

6.1 Propagation Kernels

Taking the help of propagation kernels (Neumann et al., 2015), We define a kernel $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ among graph instances $G^{(i)} \in \mathcal{X}$. The input space $\mathcal{X}$ comprises graphs $G^{(i)} = (V^{(i)}, E^{(i)}, \ell)$, where $V^{(i)}$ is the set of nodes and $E^{(i)}$ is the set of edges in graph $G^{(i)}$. Weighted adjacency matrices $A^{(i)} \in \mathbb{R}^{n_i \times n_i}$ represents and the label function ℓ endows nodes with label and attribute information. The similarity between two graphs $G^{(i)}$ and $G^{(j)}$ is to computed by comparing all pairs of nodes in the above two:

$$K(G^{(i)}, G^{(j)}) = \sum_{v \in G^{(i)}} \sum_{u \in G^{(j)}} k(u, v),$$

where $k(u, v)$ is an arbitrary node kernel determined by node labels and, if present, node attributes. It is defined in terms of the nodes' corresponding probability distributions $p_{t,u}$ and $p_{t,v}$, which we update and maintain throughout the process of information propagation.

The kernel contribution of iteration t is defined by

$$K(G_t^{(i)}, G_t^{(j)}) = \sum_{v \in G_t^{(i)}} \sum_{u \in G_t^{(j)}} k(u, v). \tag{6.4}$$

The propagation kernels between labeled and attributed graphs are defined as:

$$k(u, v) = k_l(u, v) \cdot k_a(u, v), \tag{6.5}$$

where $k_l(u, v)$ is a kernel corresponding to label information and $k_a(u, v)$ is a kernel corresponding to attribute information. If no attributes are present, then $k(u, v) = k_l(u, v)$. The t_{MAX}-iteration propagation kernel is now given by

$$K_{t_{\text{MAX}}}(G^{(i)}, G^{(j)}) = \sum_{t=1}^{t_{\text{MAX}}} K(G_t^{(i)}, G_t^{(j)}). \tag{6.6}$$

If node kernel is defined in the form

$$k(u, v) = \begin{cases} 1 & \text{if } \textit{condition} \\ 0 & \text{otherwise,} \end{cases} \tag{6.7}$$

where *condition* is an equality condition on the information of nodes u and v, we can compute K efficiently by *binning* the node information, *counting* the respective bin strengths for all graphs, and *computing a base kernel* among these counts. That is, we compute count features $\phi(G_t^{(i)})$ for each graph and plug them into a base kernel: $\langle \cdot, \cdot \rangle$

Algorithm 1 The general propagation kernel computation (Neumann et al., 2015).

```
given: graph database $\{G^{(i)}\}_i$, # iterations $t_{\text{MAX}}$, propagation scheme(s), base kernel $\langle \cdot, \cdot \rangle$
$K \leftarrow 0$, initialize distributions $P_0^{(i)}$
for $t \leftarrow 0 \dots t_{\text{MAX}}$ do
    for all graphs $G^{(i)}$ do
        for all nodes $u \in G^{(i)}$ do
            quantize $p_{t,u}$, where $p_{t,u}$ is $u$-th row in $P_t^{(i)}$        ▻ bin node information
        end for
        compute $\Phi_{i\cdot} = \phi(G_t^{(i)})$        ▻ count bin strengths
    end for
    $K \leftarrow K + \langle \Phi, \Phi \rangle$        ▻ compute and add kernel contribution
    for all graphs $G^{(i)}$ do
        $P_{t+1}^{(i)} \leftarrow P_t^{(i)}$        ▻ propagate node information
    end for
end for
```

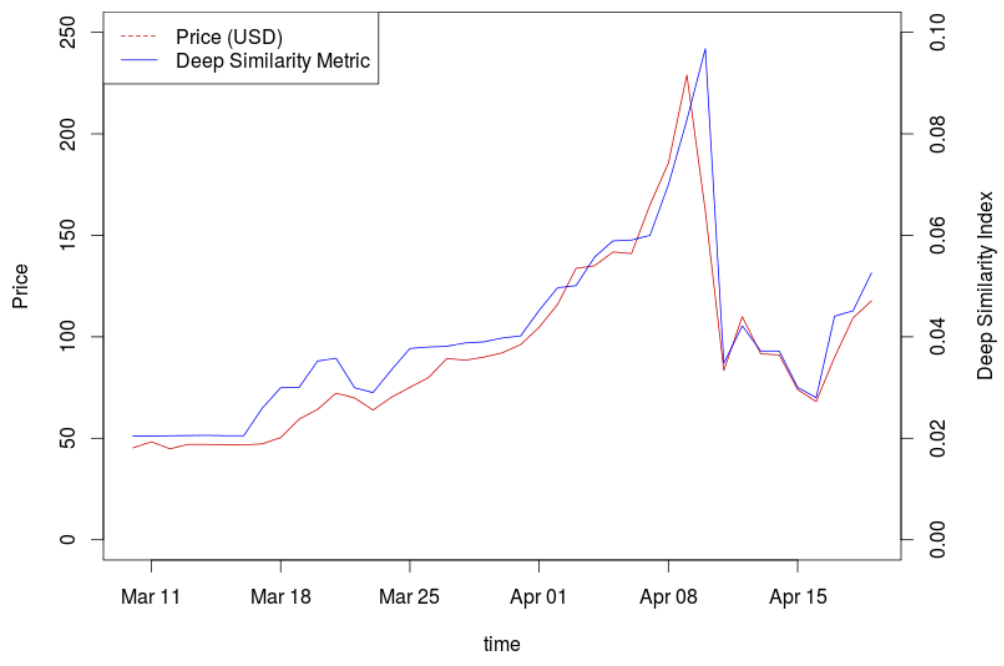
$$K(G_t^{(i)}, G_t^{(j)}) = \langle \phi(G_t^{(i)}), \phi(G_t^{(j)}) \rangle. \tag{6.8}$$

The above mathematical equations can be put in the form of algorithms 1.

6.2 Experiments

We compare our framework with the representative instances of major families of graph kernels as discussed in Yanardag and Vishwanathan (2015). The similarity index is calculated as discussed in section 5. The Matlab code of the propagation kernels was obtained from (Neumann et al., 2015), which was then coded in python. The correlation between transaction network structure and market price is depicted by figure 6.1. It shows better correlation with the network, as compared to similarity index calculated by other graph kernels without using deep framework.

Figure 6.1 Deep Similarity



7 Future Directions

In this report, we find strong correlation between structural property (walk) of bitcoin transaction graphs and BTC/USD exchange rate (price) by calculating similarity index using basic random walk graph kernels(Gärtner et al., 2003; Kashima et al., 2003).

As we find strong correlations ($\rho = 7.7$) above, the same experiments is expected to performed taking other robust graph kernels in particular shortest path, free from tottering and halting problems. We intend to use "propagation kernels" (Neumann et al., 2015), which can be used to construct kernels for many graph types, including labeled, partially labeled, unlabeled, directed, and attributed graphs, considering we have weighted dircted labled attributed graph. This makes easy to extend our work taking account of attributes to do price forecast in future.

Since, (Yanardag and Vishwanathan, 2015) deep graph kernels outperforms popular graph kernels, we intent to use the same approach with modification in graph sampling and adding other graph kernels, especially propagation kernels (Neumann et al., 2015).The deep graph propagation kernels outperforms other graph kernels without deep framework, which can be seen in the figure 6.1.

Though, we wanted to reproducing results from Kondor et al. (2014) studies, but that will depend on the time. But, we will use Lee et al. (2015) graph similarity approach, based on random walk with restart (RWR) algorithm with intergraph compression to transform representation of graph for comparing it with graph kernels or its variant.

References

F. Aziz, R.C. Wilson, and E.R. Hancock. Backtrackless walks on a graph. *Neural Networks and Learning Systems, IEEE Transactions on*, 24(6):977–989, June 2013.

László Babai. Graph isomorphism in quasipolynomial time. *CoRR*, abs/1512.03547, 2015. URL http://arxiv.org/abs/1512.03547.

K.M. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. In *Data Mining, Fifth IEEE International Conference on*, pages 8 pp.–, Nov 2005.

Mukund Deshpande, Michihiro Kuramochi, Nikil Wale, and George Karypis. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Trans. on Knowl. and Data Eng.*, 17(8):1036–1050, August 2005.

Aasa Feragen, Niklas Kasenburg, Jens Petersen, Marleen de Bruijne, and Karsten M. Borgwardt. Scalable kernels for graphs with continuous attributes. In *NIPS*, pages 216–224, 2013.

Michael Fleder, Michael S. Kester, and Sudeep Pillai. Bitcoin transaction graph analysis. *CoRR*, abs/1502.01657, 2015. URL http://arxiv.org/abs/1502.01657.

Thomas Gärtner, Peter Flach, and Stefan Wrobel. *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003. Proceedings*, chapter On Graph Kernels: Hardness Results and Efficient Alternatives, pages 129–143. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

Johannes Göbel, H. Paul Keeler, Anthony E. Krzesinski, and Peter G. Taylor. Bitcoin blockchain dynamics: the selfish-mine strategy in the presence of propagation delay. *CoRR*, abs/1505.05343, 2015. URL http://arxiv.org/abs/1505.05343.

David Haussler. Convolution kernels on discrete structures. Technical Report UCS-CRL-99-10, University of California at Santa Cruz, Santa Cruz, CA, USA, 1999. URL http://citeseer.ist.psu.edu/haussler99convolution.html.

U. Kang, Hanghang Tong, and Jimeng Sun. Fast random walk graph kernel. 2012.

Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 321–328. AAAI Press, 2003.

Gnter Klambauer, Martin Wischenbart, Michael Mahr, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Rchemcpp: a web service for structural analoging in chembl, drugbank and the connectivity map. *Bioinformatics*, 31(20):3392–3394, 2015.

Dániel Kondor, István Csabai, János Szüle, Márton Pósfai, and Gábor Vattay. Inferring the interplay between network structure and market effects in bitcoin. *New Journal of Physics*, 16(12):125003, 2014.

Danai Koutra, Joshua T. Vogelstein, and Christos Faloutsos. DELTACON: A principled massive-graph similarity function. *CoRR*, abs/1304.4657, 2015.

Ladislav Kristoufek. What are the main drivers of the bitcoin price? evidence from wavelet coherence analysis. *PLoS ONE*, 10(4), 04 2015.

Jaekoo Lee, Gunn Kim, and Sungroh Yoon. Measuring large-scale dynamic graph similarity by ricom: RWR with intergraph compression. In *2015 IEEE International Conference on Data Mining, ICDM 2015, Atlantic City, NJ, USA, November 14-17, 2015*, pages 829–834, 2015.

Geng Li, Murat Semerci, Bülent Yener, and Mohammed J. Zaki. Effective graph classification based on topological and label attributes. *Stat. Anal. Data Min.*, 5(4):265–283, August 2012.

Stephanie Lo and J. Christina Wang. Bitcoin as money? Current Policy Perspectives 14-4, Federal Reserve Bank of Boston, 2014.

Pierre Mahé, Nobuhisa Ueda, Tatsuya Akutsu, Jean-Luc Perret, and Jean-Philippe Vert. Extensions of marginalized graph kernels. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 70–, New York, NY, USA, 2004. ACM.

Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of the 18th International Conference on Data Engineering*, ICDE '02, pages 117–, Washington, DC, USA, 2002. IEEE Computer Society.

A. Mheich, M. Hassan, V. Gripon, M. Khalil, C. Berrou, O. Dufor, and F. Wendling. A novel algorithm for measuring graph similarity:

Application to brain networks. In *Neural Engineering (NER), 2015 7th International IEEE/EMBS Conference on*, pages 1068–1071, April 2015.

Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. May 2009. URL http://www.bitcoin.org/bitcoin.pdf.

Marion Neumann, Roman Garnett, Christian Bauckhage, and Kristian Kersting. Propagation kernels: efficient graph kernels from propagated information. *Machine Learning*, 102(2):209–245, 2015.

Francesco Orsini, Paolo Frasconi, and Luc De Raedt. Graph invariant kernels. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 3756–3762. AAAI Press, 2015. URL http://ijcai.org/papers15/Abstracts/IJCAI15-528.html.

Dorit Ron and Adi Shamir. Quantitative analysis of the full bitcoin transaction graph. In Ahmad-Reza Sadeghi, editor, *Financial Cryptography*, volume 7859 of *Lecture Notes in Computer Science*, pages 6–24. Springer, 2013.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-lehman graph kernels. *J. Mach. Learn. Res.*, 12:2539–2561, November 2011.

Michele Spagnuolo. Bitiodine: Extracting intelligence from the bitcoin network. Master's thesis, Politecnico di Milano, Italy, 2013.

Mahito Sugiyama and Karsten Borgwardt. Halting in random walk kernels. In *Advances in Neural Information Processing Systems 28*, pages 1630–1638. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5688-halting-in-random-walk-kernels.pdf.

S V N Vishwanathan, Karsten M. Borgwardt, and Nicol Schraudolph. Fast computation of graph kernels. In *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.

S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. Graph kernels. *J. Mach. Learn. Res.*, 11:1201–1242, August 2010.

Pinar Yanardag and S.V.N. Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1365–1374, New York, NY, USA, 2015. ACM.

Laura A. Zager and George C. Verghes. Graph similarity scoring and matching. *Applied Mathematics Letters*, 21(1):86 – 94, 2008.