Introduction

Graphs are a ubiquitous data structure. With the computer science it's a very commonly related field which it can related with the further nodes or object. For this reason using of the graph are increasing day by day. Like social media, Biological protein-protein networks, recommender systems all of these are related to the graph system [1]. There are so may filed and research on it which try to implementation this graph idea. One main thing that when we talk about the graph we may be think that it may be the structural knowledge, graph are not only for graph repositories but also it's play the important role in machine learning. For example , classify the role of the protein in biological interaction graph [2], recommended new friends in social media [3], predict new medicine for the application of existing drug molecules [4].

There are so many ways to implement this project and it's discussion area is too large. To solve this problem there are so many algorithms. And the most popular algorithms are:

1. DeepWalk (inspired by word2vec in NLP main goal is node embedding)
2. Graph Convolutional Networks (GCN)
3. Graph SAGE (Sample and aggreGatE)
4. ChebNet (CNN on graphs with fast localized spectral filtering)
5. Graph Attention Networks

In this project we are implement one algorithm for further research and development we try to use other algorithm. To implement the algorithm we must to realize the problem. Motivation of this project-

Let, Given a graph "G". Convert the nodes into embedding vector. Nodes are connected/Adjacency matrix. No initial feature matrix to the data. We construct a feature matrix which will have all randomly selected nodes. It may be multiple method but first we assuming it normally sample.

To solve this we can use many algorithm. But in this project we used RandomWalk algorithm because of think of the complexity and time limitation of the project.

RandomWalk:

Think that the rooted at vertex v_i as w_{v_i}. Stochastic process with random variable W_{vi}^1 , W_{vi}^2, … … … … W_{vi}^k such that W_{vi}^{k+1} -> vertex chosen at random from the neighbors of vertex v_k

PowerLaw:

RandomWalk algorithm are scale free networks. Then main question to implement this problem is Why it is important? To answer this simple question we must be need to read full paper then we think we can understand this question answer.

SkipGram:

- Learning a representation in word2vec means learning a mapping function from the word co-occurrences & that is exactly what we are heading for.
- Assume probability => p(x|y) ; y= set of the words that appear in the same sentence which x occurs
- Input layer have |v| neurons
- |v| number of the words that are interesting to us.
- Using one hidden layer for simplicity.
- It can have as many neurons if than the number of word in vocabulary.
- Output layer also |v| neurons.
- Interpretation of input & output layer (Now we don't care about the hidden layer)
- Suppose vocabulary v1,v2, … … … vi, … … …vn we try to maximize probability v4, v7, v9
- Input layer 4^{th}, 7^{th} ,9^{th} with value 1 others with value 0
- Hidden layer have some functions
- Hidden layer has no non-linear activation
- Apply sigmoid and get probability

DeepWalk = SkipGram Analogy + Random Walks

Random Walks:

- Think of as short sentence & phases in special language.
- The direct analogy is to estimate observing vertex v_i , all previous vertices visited so far in the random walk
- **Goal:** learn a latent representation, not only probability distribution of node co-occurrences so we introduce mapping function.

$$\Phi : v \in V \rightarrow R^{|v| \times d}$$

Matrix of free parameters which serve later on as our X_E

- The problem then is to estimate the likelihood ->
-

$$P_r(v_i \mid \Phi(v_1), \Phi(v_2), \ldots \ldots, \Phi(v_{i-1}))$$

- DeepWalk algorithm uses notation of Random Walks to get the surrounding nodes/words and ultimately calculate the probability given the context node.
- It start with one node and finds out all the nodes which have an edge connecting with the start node repeat this procedure.
- After n iterations we have traversed n nodes(some may be repeat, but it doesn't matter)

- N-nodes as the surrounding nodes for the original node, will try to maximize probability using the probability estimate.

DeepWalk Algorithm:

DeepWalk (G, w, d, δ, t)

Input: graph G (V, E)

Window size w

Embedding size d

Walks per vertex δ

Walk length t

Output: matrix of vertex representation Φ ∈ $R^{|v|\times d}$

1. Initialization : Sample Φ from $u^{|v|\times d}$
2. Build a binary tree T from V
3. For i = 0 to δ do

   Θ = shuffle (v)

   For each v_i ∈ Θ do

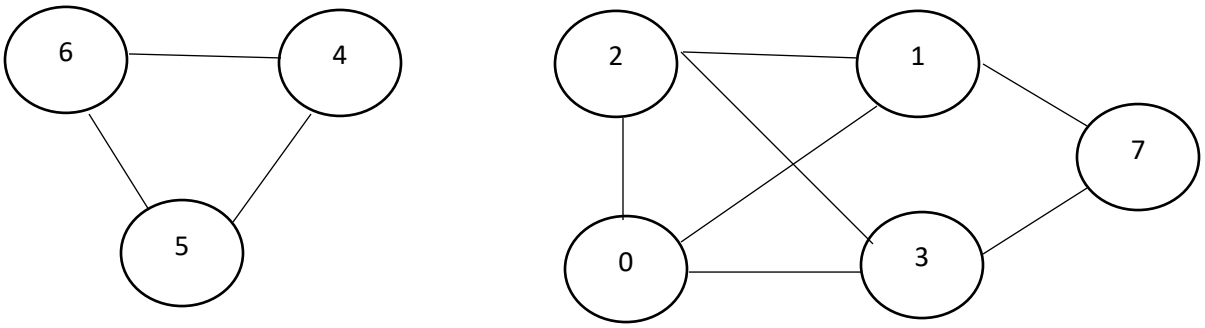
   w_{v_i} = RandomWalk (G, v_i, t)

   Skipgram (Φ, w_{v_i} , w)

   End for

   End for

Example to implement DeepWalk



SkipGram:

- Closely related to the CBOW model
- Maximize the probability of the word given it's surrounding using Neural Network
- Weight learned from the input to the hidden layer
- It has been some problem |v| vocabulary for each iteration, we will be modifying a total of |v| vectors. It would be complex because there will be millions of vocabulary.
- To solve this problem we need to use some method like Hierarchical Softmax or negative sampling to reduce complexity.

Hierarchical Softmax:

- Probability of any outcome depends on the total outcomes of our model.
- Probability depends on the number of model parameters via the normalization constant in the softmax

SoftMax $(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$

- No parameter are linear in the total number of outcome.
- We are dealing with huge graphical structure, it's costly & taking much time.

Overcome Challenge:

- Hierarchical probability of any outcome depends on several model parameters that is only logarithmic in the total number of outcomes
- Uses binary three to represent all words/nodes in the vocabulary.
- Probability distribution from O(v) to O (log(v))

0020

Reference :

[1]

[2]W.L. Hamilton, R. Ying, and J. Leskovec. Inductive representation Learning on Large graphs. arXiv preprint ,2017.

[3]L. Backstrom and J. Leskovec. Supervised random walks: predicting and recommending links in social networks. In WSDM, 2011.

[4]D. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R.P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In NIPS, 2015.