University of
Cambridge

Mathematics Tripos

Part III Essay

Walking Deeper on Dynamic Graphs

December 13, 2019

Written by
Joshua Snyder

Contents

1 Motivation

The motivation for studying nodes in graphs and their representations comes from the desire to understand networks of people or objects and their relations. The motivating question for this essay is

> **Question** (Motivating Question)**.** Given a large network of people how can we quantify the relationships between them?

The natural way to go about this is to let nodes represent persons and let edges between them represent connections from which we can induce some understanding of relationship or trust between two people. This task is difficult; even with a relatively small number of people it is not a task on which humans perform very well and the task rapidly becomes difficult as the number of nodes in the graph increases.
Many of the networks which we wish to study are dynamic; that is they change with time. In a large network it is common that small changes occur during each epoch of time that over time cause larger changes to the network structure. How do we quantify these changes and their impact on the relationships in the network without having to completely re-analyse the network after ever epoch of time, losing the information that we already learned. A frequent example upon which we can draw similarity is that of social networks. A large social network has new users (nodes) being added and new relationships (edges) formed in each epoch of time, however in any given short time period the graph representing the social network does not change substantially. It would be both foolish and costly to re-analyse the graph after each epoch however most of the previous literature has focused on static graphs. In the latter half of this essay I will give an account of recent progress into the application of DeepWalk and similar algorithms to dynamic graphs.

2 Summary

There are three main objectives of this essay:

- Firstly the essay will give a mathematical outline of the DeepWalk algorithm and it's application to social representation learning. We will briefly discuss the benefits and drawbacks of the algorithm.
- Secondly we will look at an implementation of DeepWalk to dynamic graphs. The challenge here is to develop an unbiased representation of a graph at time $t+1$ given it's representation at time t, without re-analysing the entire network. This is a very important task since many networks, especially social ones, are constantly changing. However in any given epoch of time the network structure is unlikely to undergoe dramatic change and so a computationally effective algorithm will not re-analyse the network at each step.
- Thirdly, we will exhibit an implementation of the outlined dynamic DeepWalk application to a social data set [which data set?] and give suggestions as to good applications of Dynamic DeepWalking. [talk here about the application once I have found one]

3 DeepWalk

This section gives an outline of the social representation learning algorithm DeepWalk, first introduced in the seminal paper DeepWalk: Online Learning of Social Representations by B. Perozzi et. al. [1]. The method proposed in this paper not only demonstrated performance improvements from previous methodologies but also motivated an entirely different approach. At the time of the paper being written, significant advancements were being made in natural languade processing (NLP) and the idea of word embeddings was becoming popular through an embedding algorithm known as word2vec [2, 3]. DeepWalk implements this algoritm but replaces the idea of the context of a word in a sentence with the context of a node in a random walk on a graph. This is the crucial concept of DeepWalk from which the remaining details naturally follow.

The original paper on DeepWalk is lacking in a mathematical underpinning and in this section we will model the algorithm mathematically. It is suggested that the reader is familiar with the concepts outlined in the paper by Perozzi et. al. prior to reading this (more) mathematical exposition. I have endevoured to use similar notation to the original paper to ease cross-referencing. Without further ado, let us begin our journey.

> **Definition.** Let $G = (V, E)$ be an undirected graph (representing a network). V represents the members of the network, commonly referred to as the nodes and $E \subset V \times V$ represents their connections.
>
> The nodes and edges have lables and $G_L = (V, E, X, Y)$ represents the partially labelled network. $X \in \mathbb{R}|V| \times S$ where S is the size of the feature space for each attribute vector and $Y \in \mathbb{R}|V| \times |\mathcal{Y}|$, where $\mathcal{Y}$ is the set of labels.

Our goal is to learn $X_E \in \mathbb{R}^{|V| \times d}$ where d is a small number of latent dimensions. The idea is that each latent dimension contributes a dimensional

3.1 SkipGram

To understand DeepWalk mathematically, we first need to understand what the SkipGram model is doing, since this is the model underpinning DeepWalk, which took it from NLP and applied it to graphs. It is easiest to first gain an understanding of SkipGram in it's original context.

Paragraph with word analogies: Skipgram trains a neural network to do the following task. Given an input word, located somewhere in a sentence, pick a nearby word at random. The task of the network is to give the probability of each word in the vocabulary being the nearby word that we choose. As an example, if the context work is "rainy" then "weather" would be assigned a high probability whereas "carrot" would be assigned a low one.

Paragraph without word analogies: In the context of graph networks, SkipGram trains a neural network to do the following task.
Given an input vertex v and a random walk, W_v, of length $2t + 1$ with v at it's centre. Pick a nearby vertex at random. The task of the neural network is to predict the probability that each vertex in V will be the randomly chosen

vertex. Therefore, verticies far away on the graph that correspond to unfamiliar nodes are unlikely to co-occur on the same random walk and will be assigned a low probability. Conversely, nearby and well connected vertices are likely to co-occur on a random walk with input v and thus will be assigned higher probabilities. This allows us to train a network with weights that represent the connectedness between nodes on the graph.

The neural network is trained by feeding it pairs of nodes (v, w) where v represents the input node and w is a context node, which lies within a certain window size.

To formalise this, each of the nodes $v \in V$ are represented by a one-hot encoding vector $e_v \in \mathbb{R}|V|$ allowing us to feed e_v into the neural network. When e_v is fed into the network, a single linear hidden layer with d neurons is used, where d is the desired dimension of the latent representations, which is then passed to a softmax classifier for output. The output of the network is a vector $o \in \mathbb{R}d$ containing the estimated probabilities that a randomly selected nearby word is that vocabulary word.

The idea behind having a linear hidden layer, which does not use an activation function, is to use the resulting weight matrix $W \in \mathbb{R}|V| \times d$ as the embedding vectors for the nodes in the graph. This is intuitive as the hidden layer acts as a bottleneck that tries to represent as much information as possible to distinguish the nodes, but is only allowed d neurons to do so. Since $d \ll |V|$ there is a low risk of overfitting.

*** Is our vocabulary the whole graph, or is it just the random walk that we feed to the SkipGram model, perhaps it is just the random walk, be careful with this! ***

*** Create here an image resembling the one found in the blog post on SkipGram but for graphs and with d dimensions ***

The algorithm used in DeepWalk varies slightly from the SkipGram algorithm discussed here in two major ways. Firstly, calculating the normalization factor in the Softmax layer requires a computational complexity of $O(|V|)$, this is reduced by using Hierarchical Softmax to approximate the Softmax probabilities, requiring a complexity of only $O(log|V|)$. In particular, a Huffman coding is used to reduce the access time of frequent elements in the tree, as suggested by Mikolov et al. in the original Word2Vec papers.[3, 2]

Secondly, in the implementation of Word2Vec (and hence DeepWalk), Negative Sampling is used.[3] This was shown by Levy and Goldberg[4] to be implicitly factorising a matrix M. The proof is presented here in the context of graph representations.

Theorem 3.1 (Levy, Goldberg (2014))**.** *SkipGram with Negative Sampling (SGNS) is implicitly factorising the matrix*

$$M = \log \frac{\#(v,c)|\mathcal{D}|}{\#(v)\#(c)} - log$$

$b= WCT \in \mathbb{R}|V| \times |V|$ *where* $W \in \mathbb{R}|V| \times d$ *and* $|C| \in \mathbb{R}d \times |V|$ *and* b *is the number of negative samples.*

In the above theorem, $\mathcal{D}$ represents the random walk corpus and $\#(v, c)$, $\#(v)$ and $\#(c)$ denote the number of times vertex-context pair (v, c), vertex v and context c appear in the corpus respectively.

Proof. Exhibit the proof from Levy Goldberg. □

4 Matters of Convergence

5 DeepWalk Sucks

6 Dynamic Deep Walking

7 Discussion of an application

Essay Descriptor

Walking Deeper on Dynamic Graphs: Learning Latent Representations with Random Walks for Image Classification

In the era of big data, graph representation is a natural and powerful tool for representing big data in real-world problems [1],[2],[4]; some examples include data coming from medical records, social networks, recommendation systems and transport systems. A challenging question when using graph representation is how to learn latent representations on multi-label networks for several classification tasks, and a seminal algorithm for this is the DeepWalk technique using random walks [1].

We propose two questions for investigation in this essay. Firstly, we hope that students will develop a rigorous mathematical underpinning for the DeepWalk algorithm, in the spirit of convergence guarantees.

Secondly, we seek to investigate the connection of DeepWalk to dynamic graphs. Many realworld events are dynamic - for example, in a social network new users are constantly added- while most of the body of literature is based on the unrealistic assumption that the graph is static. From the learning point of view, this assumption has a negative impact in the computations, as the graph has to be re-learned each time that an instance changes. We also hope that students will also discuss some open questions that they find interesting.

Relevant Courses

Useful: Background knowledge in Machine Learning and Statistics is helpful, as is probability to the level of Part II Applied Probability. Some content from Part III Mixing Times of Markov Chains, on the long-time behaviour of random walks on graphs, may also be useful.

References

[1] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 03 2014.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[3] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," 2013.

[4] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," *Advances in Neural Information Processing Systems*, vol. 3, pp. 2177–2185, 01 2014.