Notes on:
DeepWalk - Online Learning of Social Representations

August 29, 2024

1 Abstract

DeepWalk was presentes as a novel approach for learning representation of vertices in networks. This is done by random walks to learn representations and converting them to sentences (embeddings). The increase in F1 scores was up to 10% compared to other methods for social networks like Youtube.

2 Introduction

Machine learning for networks must be able to cope with the sparsity of network classification, anomaly detection and missing link prediction. DeepWalk generalises neural language models to process a special language composed of a set of randomly-generated walks. It takes a Graph as an input and produces a latent representation as output. To demonstrate potential of this method, it was utilised in challenging multi-label network classification in large heterogeneous graphs.

Contributions:

1. Introduce deep learning as a tool to analyse graphs. DeepWalk learns structural features.
2. Evaluated results with a 5-10% improvement compared to toher methods.
3. Demonstrated the scalability of the algorithm by using Youtube's big scale data/graphs.

3 Problem definition

Let $G = (V, E)$, where V are the memebers of the network (nodes or vertices) and E its edges, $E \subseteq (V \times V)$ (A subset of the cartesian product of V) Given a partially labeled social network $G_L = (V, E, X, Y)$, with attributes $X \in \mathbb{R}^{|V| \times S}$

where S is the size of the feature space for each attribute vector, and $Y \in \mathbb{R}^{|V|\times|\mathcal{Y}|}$, $\mathcal{Y}$ is the set of labels.

In traditional machine learning, we aim to learn H that maps elements of X to the set of labels $\mathcal{Y}$. DeepWalk uses the embedded information in the structure G to achieve superior performance.

This method instead of mixing the label space as part of the feature space, an unsupervised method which learns the graph structure independent to the labels.

4 Random Walks

A random walk is rooted at vertex v_i as $\mathcal{W}_{v_i}$. It is a stocastic process with random variables $\mathcal{W}_{v_i}^1, \mathcal{W}_{v_i}^2, \ldots, \mathcal{W}_{v_i}^k$ such that $\mathcal{W}_{v_i}^{k+1}$ is a vertex chosen at random from the neighbors of vertex v_k.

5 Connnection: Power laws

If the degree distribution of connected graph (the amount of edges that each vertice has) follows a power law (is scale-free (which means that it is scale-invariant, the relative proportion degree/edges remains the same doesn't matter the size of the graph)), random walks will also follow a power-law distribution.

6 Language modelling

The goal of language modeling is to estimate the likelihood of a specific sequence of words appearing in a corpus (large structured set of texts, dataset for NLP training).

$$W_1^n = (w_0, w_1, \ldots, w_n) \tag{1}$$

where $w_i \in \mathcal{V}$ ($\mathcal{V}$ is the vocabulary), we would like to maximize the $\Pr(w_n \mid w_0, w_1, \ldots, w_{n-1})$ (maximise the probability of w_n, after all given words that precede $(w_0, w_1, \ldots, w_{n-1})$) over corpus dataset.

In this case, the analogous to this example in graphs is to estimate the likelihood of observing vertex v_i given all the previous vertices visited so far in a random walk.

$$\Pr(v_i \mid v_1, v_2, \ldots, v_{i-1}) \tag{2}$$

As this method's goal is to learn a latent representation, not just a probability distribution of node co-occurrences, a mapping function is introduced $\Phi : v \in V \mapsto \mathbb{R}^{|V|\times d}$. This mapping represents the latent social representation of each vertex v. Then the problem becomes:

$$\Pr(v_i \mid (\Phi(v_1), \Phi(v_2), \ldots, \Phi(v_{i-1}))) \tag{3}$$

However, as walk length grows, computing this objective function becomes unfeasible. A way to avoid this problem, instead of using the contexto to predict

a missing word, it uses one word to predict the context. The context is given by the words on the right and left side of the word. This also removes ordering constraint.

Now the model is required to maximize the probability of a word appearing in the context without the knowledge of its offset from the given word. This is done by negative log-likelihood which is commonly used as minimizing this function corresponds to maximizing the likelihood of observing the data.

$$\underset{\Phi}{\text{minimize}} \; -\log \Pr\left(\{v_{i-w}, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{i+w}\} \mid \Phi(v_i)\right) \tag{4}$$

Eq. 4 is the conditional probability of observing a set of vertices within a window of size w. If you have a linear Graph $G = v_1, v_2, \ldots, v_8$, with edges $E = \{(W_i, W_{i+1}) \mid 1 \leq i \leq n\}$. Considering v_4 as your target vertex, with a $w = 2$, this function would learn the representation of the vertices v_2 to v_6 excluding the target vertex v_4.

Solving the optimisation problem builds a representation that capture similarities in local graph structure between vertices.

7 Method

Random walk generator takes a graph G rooting at vertex v_i for the random walk $\mathcal{W}_{v_i}$. In practice, each iteration is doing a "pass" over the data and sample one walk per node during each pass.

SkipGram is an algorithm used in NLP that maximises the co-occurrence probability among the words that appear withing a window, w, in a sentence.

In this case, a SkipGram is used to iterate over all possible collocations in a random walk that appear within a window w. For each, each vertex v_j is maped into it's current representation $\Phi(v_j) \in \mathbb{R}^d$. Given the representation of v_j we want to maximise the probability of its neighbors in the walk. To speed up this training, hierarchical Softmax can be used.

8 Hierarchical Softmax

Given that $u_k \in V$, calculating $\Pr(u_k \mid \Phi(v_j))$ is not feasible (computational expensive). If the path to vertex u_k is identified by a sequence of tree nodes $\left(b_0, b_1, \ldots, b_{\lceil \log |V| \rceil}\right)$, ($b_0 = \text{root}$, $b_{\lceil log|V| \rceil} = u_k$) then

$$\Pr(u_k \mid \Phi(v_j)) = \prod_{l=1}^{\lceil \log |V| \rceil} \Pr(b_l \mid \Phi(v_j) \tag{5}$$

Now $\Pr(b_l \mid \Phi(v_j))$ is modelled by a binary classifier that is assigned to the parent of the node b_l.