图表示学习

Zhou Shen

School of Computer Science, Wuhan University

October 23, 2021

Outline

1 Word2Vec

2 DeepWalk

3 Node2Vec

4 Struc2Vec

5 LINE

6 SDNE

1-of-N Encoding

apple = [1 0 0 0 0]

bag = [0 1 0 0 0]

cat = [0 0 1 0 0]

dog = [0 0 0 1 0]

elephant = [0 0 0 0 1]

Word2Vec

One-Word Context

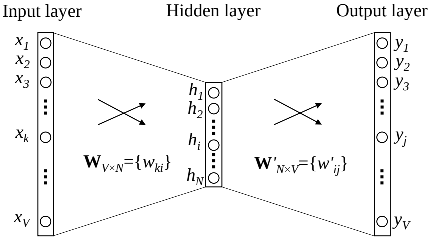


Figure 1: A simple CBOW model with only one word in the context

$$\mathbf{h} = \mathbf{W}^T\mathbf{x} = \mathbf{W}^T_{(k,\cdot)} := \mathbf{v}^T_{w_I}, u_j = \mathbf{v}'_{w_j}h \tag{1}$$

$$p\left(w_j \mid w_I\right) = y_j = \frac{\exp\left(u_j\right)}{\sum_{j'=1}^{V} \exp\left(u_{j'}\right)} \tag{2}$$

Word2Vec

Objective Function

$$
\begin{aligned}
\max p\left(w_O \mid w_I\right) &= \max y_{j^*} \\
&= \max \log y_{j^*} \\
&= u_{j^*} - \log \sum_{j'=1}^{V} \exp\left(u_{j'}\right) := -E
\end{aligned} \tag{3}
$$

常用变形

$\min E = -u_{j^*} + \log \sum_{j'=1}^{V} \exp\left(u_{j'}\right)$

Word2Vec

CBOW

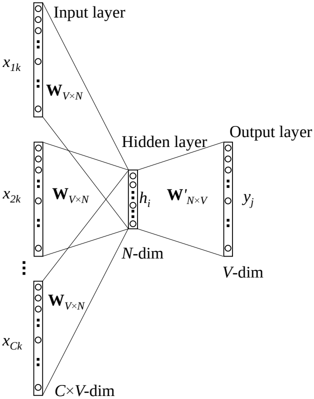


Figure 2: Continuous bag-of-word model

$$\mathbf{h} = \frac{1}{C}(\mathbf{v}_{w_1} + \mathbf{v}_{w_2} + \cdots + \mathbf{v}_{w_C})^T$$

$$\begin{aligned} E &= -\log p\left(w_O \mid w_{I,1}, \cdots, w_{I,C}\right) \\ &= -u_{j^*} + \log \sum_{j'=1}^{V} \exp\left(u_{j'}\right) \end{aligned}$$

Word2Vec

Skip-Gram

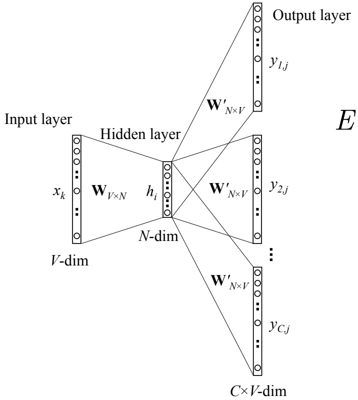
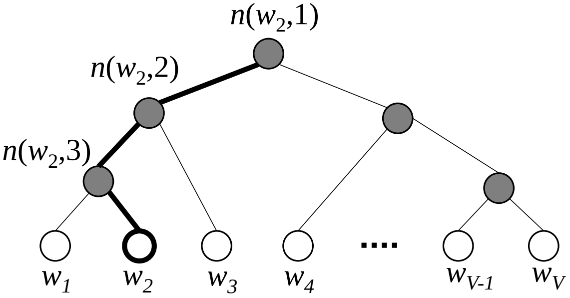


Figure 3: The skip-gram model.

$$
\begin{aligned}
E &= -\log p\left(w_{O,1}, w_{O,2}, \cdots, w_{O,C} \mid w_I\right) \\
&= -\log \prod_{c=1}^{C} \frac{\exp\left(u_{c,j_c^*}\right)}{\sum_{j'=1}^{V} \exp\left(u_{j'}\right)} \\
&= -\sum_{c=1}^{C} u_{j_c^*} + C \cdot \log \sum_{j'=1}^{V} \exp\left(u_{j'}\right)
\end{aligned}
$$

Word2Vec

Hierarchical Softmax



$$p\left(w = w_O\right) = \prod_{j=1}^{L(w)-1} \sigma\left(\left[n(w, j+1) = \mathrm{ch}(n(w,j))\right] \cdot {\mathbf{v}'_{n(w,j)}}^T \mathbf{h}\right)$$

Word2Vec

Negative Sample

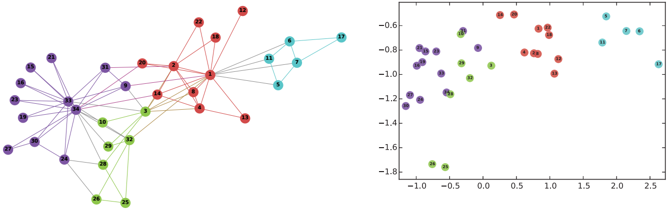
1. 词频计算

$$weight(w_i) = \frac{count(w_i)^{0.75}}{\sum_j^V count(w_j)^{0.75}}$$

2. 目标函数

$$E = -\log\sigma\left({\mathbf{v}'_{w_O}}^T\mathbf{h}\right) - \sum_{w_j\in\mathcal{W}_{\text{neg}}} \log\sigma\left(-{\mathbf{v}'_{w_j}}^T\mathbf{h}\right)$$

DeepWalk

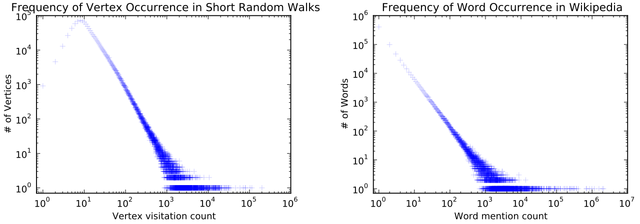


(a) Input: Karate Graph (b) Output: Representation

Definition

Problem Definition Let $G = (V, E)$, where V are the members of the network, and E be its edgs. Our goal is to learn $X_E \in \mathcal{R}^{|V| \times d}$, where d is number of latent dimensions.

DeepWalk



(a) YouTube Social Graph (b) Wikipedia Article Text

Connection

The power-law distribution of vertices appearing in the short random walks follows a power-law, much like the distribution of words in natural language.

DeepWalk

1. give a sequence of random walk from vertice i

$$\mathcal{W}_i^n = \{v_1, v_2, v_3, \cdots, v_n\} \tag{4}$$

2. maximize the likehood

$$\Pr\left(v_i \mid (\Phi(v_1), \Phi(v_2), \cdots, \Phi(v_{i-1}))\right) \tag{5}$$

Mapping Function Φ

$\Phi : v \in V \longmapsto \mathcal{R}^{|V| \times d}$ where d is the number of dimensions

Word2Vec

DeepWalk

Node2Vec

Struc2Vec

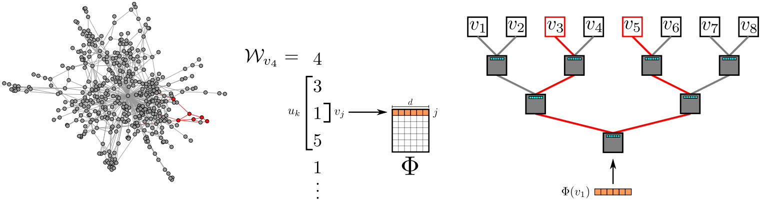
LINE

SDNE

DeepWalk

Disadvantages and Optimizations

- As the walk length grows, computing this objective function becomes unfeasible.
- Restricts the order in which vertices appear



(a) Random walk generation. (b) Representation mapping. (c) Hierarchical Softmax.

DeepWalk

Algorithm

Algorithm 1 DeepWalk(G, w, d, γ, t)

Input: graph $G(V, E)$
  window size w
  embedding size d
  walks per vertex γ
  walk length t

Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V|\times d}$

Initialization: Sample Φ from $\mathcal{U}^{|V|\times d}$
Build a binary Tree T from V
for $i = 0$ to γ **do**
  $\mathcal{O} = \text{Shuffle}(V)$
  for each $v_i \in \mathcal{O}$ **do**
    $\mathcal{W}_{v_i} = RandomWalk(G, v_i,\text{t})$
    SkipGram(Φ, $\mathcal{W}_{v_i}$, w)
  end for
end for

Corollary

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

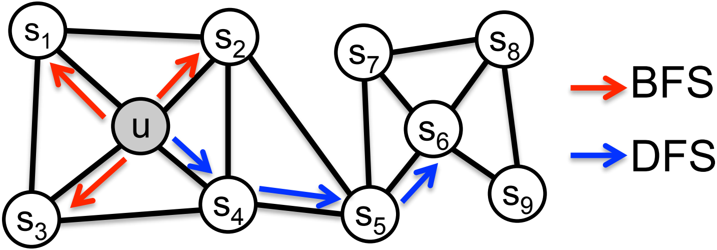
DeepWalk

复现结果

	Label Node%	10	20	30	40	50	60	70	80	90
micro-F1	原文结果	36.00	38.20	39.60	40.30	41.00	41.30	41.50	41.50	42.00
	复现结果	36.00	38.73	39.75	40.70	41.27	41.98	42.18	41.99	42.65
macro-F1	原文结果	21.30	23.80	25.30	26.30	27.30	27.60	27.90	28.20	28.90
	复现结果	20.90	23.78	25.33	26.31	27.09	27.84	28.16	28.25	28.78

Node2Vec

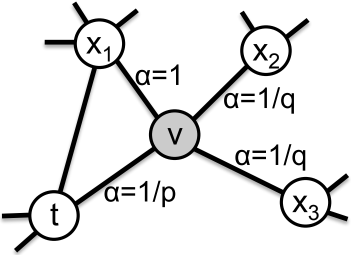
Classic Search Strategies



1. The neighborhoods sampled by BFS lead to embeddings that correspond closely to structural equivalence.
2. DFS can freely explore network neighborhoods which is important in discovering homophilous communities.

Node2Vec

Search Bias α



$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

Node2Vec

Algorithm

Algorithm 1 The *node2vec* algorithm.

```
LearnFeatures (Graph G = (V, E, W), Dimensions d, Walks per
    node r, Walk length l, Context size k, Return p, In-out q)
    π = PreprocessModifiedWeights(G, p, q)
    G' = (V, E, π)
    Initialize walks to Empty
    for iter = 1 to r do
        for all nodes u ∈ V do
            walk = node2vecWalk(G', u, l)
            Append walk to walks
    f = StochasticGradientDescent(k, d, walks)
    return f

node2vecWalk (Graph G' = (V, E, π), Start node u, Length l)
    Inititalize walk to [u]
    for walk_iter = 1 to l do
        curr = walk[−1]
        V_curr = GetNeighbors(curr, G')
        s = AliasSample(V_curr, π)
        Append s to walk
    return walk
```

Corollary

$$\max_f \sum_{u \in V} \left[\sum_{n_i \in N_S(u)} f(n_i) \cdot f(u) - \log Z_u \right]$$

Struc2Vec

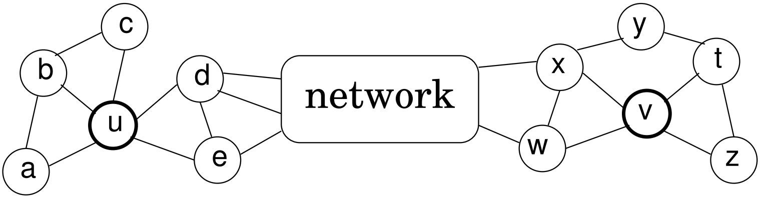


Figure 1: An example of two nodes (u and v) that are structurally similar (degrees 5 and 4, connected to 3 and 2 triangles, connected to the rest of the network by two nodes), but very far apart in the network.

Limitation

Structurally similar nodes will never share the same context if their distance (hop count) is larger than the Skip-Gram window.

Struc2Vec

Structural Similarity

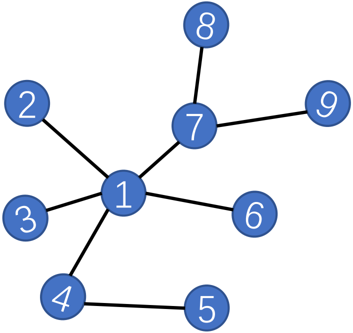
$$f_k(u,v) = f_{k-1}(u,v) + g\left(s\left(R_k(u)\right), s\left(R_k(v)\right)\right)$$
$$k \geq 0 \text{ and } \left|R_k(u)\right|, \left|R_k(v)\right| > 0 \tag{6}$$

符号解释

$f_k(u,v)$	structural distance when consider k-hop neighborhoods
$R_k(u)$	the ring of nodes at distance k
$s(S)$	the ordered degree sequence
$g(a,b)$	DTW Function

Struc2Vec

Example



Example

$$R_1(1) = \{2, 3, 4, 6, 7\}$$
$$R_2(1) = \{5, 8, 9\}$$
$$s(R_1(1)) = \{1, 1, 1, 2, 3\}$$
$$s(R_2(1)) = \{1, 1, 1\}$$

Struc2Vec

DTW

Dynamic Time Warping

利用动态规划的思想来计算两条不等长序列的相似度

动态转移方程如下：

$$g(i,j) = min\{g(i-1,j), g(i,j-1), g(i-1,j-1)\} + d(i,j)$$

$$d(a,b) = \frac{max(a,b)}{min(a,b)} - 1$$

LINE

Overview

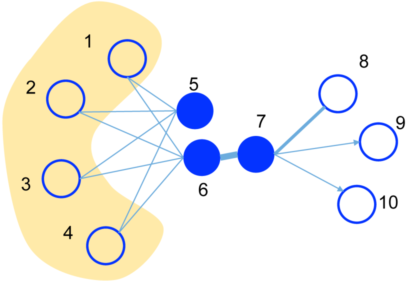


Figure 1: A toy example of information network. Edges can be undirected, directed, and/or weighted. Vertex 6 and 7 should be placed closely in the low-dimensional space as they are connected through a strong tie. Vertex 5 and 6 should also be placed closely as they share similar neighbors.

LINE

First-Order Proximity

The Joint Probability between v_i and v_j:

$$p_1\left(v_i, v_j\right) = \frac{1}{1 + \exp\left(-\vec{u}_i^T \cdot \vec{u}_j\right)}$$

Empirical Probability $\hat{p}_1(i, j) = \frac{w_{ij}}{W}$

$$O_1 = d\left(\hat{p}_1(\cdot, \cdot), p_1(\cdot, \cdot)\right)$$

where $d(\,\cdot\,,\,\cdot\,)$ is the distance between two distributions.

Objective Function

$\min\ O_1 = -\sum_{(i,j)\in E} w_{ij} \log p_1\left(v_i, v_j\right)$

LINE

Second-Order Proximity

The Probability of Context v_j generated by vertex v_i as:

$$p_2\left(v_j \mid v_i\right)=\frac{\exp\left(\vec{u}_j^T \cdot \vec{u}_i\right)}{\sum_{k=1}^{|V|} \exp\left(\vec{u}_k'^T \cdot \vec{u}_i\right)}$$

Empirical Probability $\hat{p}_1(v_j \mid v_i)=\frac{w_{ij}}{d_i}$

Objective Function

$O_2=-\sum_{(i,j)\in E} w_{ij} \log p_2\left(v_j \mid v_i\right)$

LINE

Optimization

1. Negative Sample

$$\log \sigma \left(\vec{u}'^{T}_{j} \cdot \vec{u}_i \right) + \sum_{i=1}^{K} E_{v_n \sim P_n(v)} \left[\log \sigma \left(-\vec{u}^{T}_{n} \cdot \vec{u}_i \right) \right]$$

where $\sigma(x) = \frac{1}{1+exp(-x)}$ is the sigmoid function.

2. Edge Sample - Alias Sample

SDNE
Overview

Graph Embedding

S. Zhou

Word2Vec

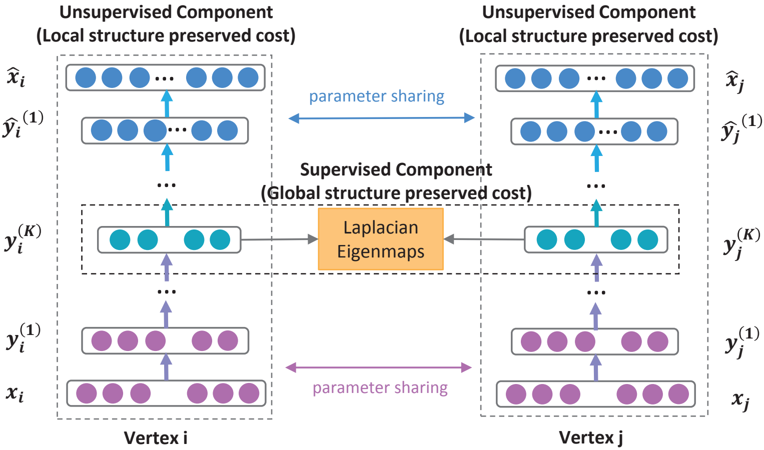
DeepWalk

Node2Vec

Struc2Vec

LINE

SDNE



SDNE

First-Order Proximity

Encoder and Decoder

$$\mathbf{y}_i^{(1)} = \sigma\left(W^{(1)}\mathbf{x}_i + \mathbf{b}^{(1)}\right)$$
$$\mathbf{y}_i^{(k)} = \sigma\left(W^{(k)}\mathbf{y}_i^{(k-1)} + \mathbf{b}^{(k)}\right), k = 2, \ldots, K$$

First-Order Proximity Loss Function

$$\begin{aligned}\mathcal{L}_{1st} &= \sum_{i,j=1}^{n} s_{i,j} \left\|\mathbf{y}_i^{(K)} - \mathbf{y}_j^{(K)}\right\|_2^2 \\ &= \sum_{i,j=1}^{n} s_{i,j} \left\|\mathbf{y}_i - \mathbf{y}_j\right\|_2^2\end{aligned}$$

SDNE

Second-Order Proximity

The autoencoder goal is to minimize:

$$\mathcal{L} = \sum_{i=1}^{n} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_2^2$$

Structural Similarity $x_i = s_i$

$$\begin{aligned}\mathcal{L}_{2nd} &= \sum_{i=1}^{n} \|(\hat{\mathbf{x}}_i - \mathbf{x}_i) \odot \mathbf{b_i}\|_2^2 \\ &= \|(\hat{X} - X) \odot B\|_F^2\end{aligned}$$

SDNE

Objective Function

正则化部分

$$\mathcal{L}_{reg} = \frac{1}{2}\sum_{k=1}^{K}\left(\left\|W^{(k)}\right\|_F^2 + \left\|\hat{W}^{(k)}\right\|_F^2\right)$$

损失函数

$$\begin{aligned}\mathcal{L}_{mix} &= \mathcal{L}_{2nd} + \alpha\mathcal{L}_{1st} + \nu\mathcal{L}_{reg} \\ &= \|(\hat{X} - X) \odot B\|_F^2 + \alpha\sum_{i,j=1}^{n} s_{i,j}\left\|\mathbf{y}_i - \mathbf{y}_j\right\|_2^2 + \nu\mathcal{L}_{reg}\end{aligned}$$