Graph Data Mining and Deep Learning

Heterogeneous Data Fusion with Multi-Layered Graphs & Graph Embedding

Nagiza F. Samatova, samatova@csc.ncsu.edu
Professor, Department of Computer Science
North Carolina State University

NC STATE UNIVERSITY
Department of Computer Science

Graph Embedding: A Way to Capture Relation Information in a Vector Space, $\mathbb{R}^d$

- (Vague) Definition: Graph embedding Φ is a mapping of a graph G into n-dimensional vector space $\mathbb{R}^n$
- (in “layman terms”): Graph embedding is a vector representation of a graph

Pattern Detection with Graph Embedding

To capture first-order and higher-order relationships via edges or links, paths or random walks

To enable the use of traditional ML techniques to mine such relationships

Data

Graph-based representation of the Data

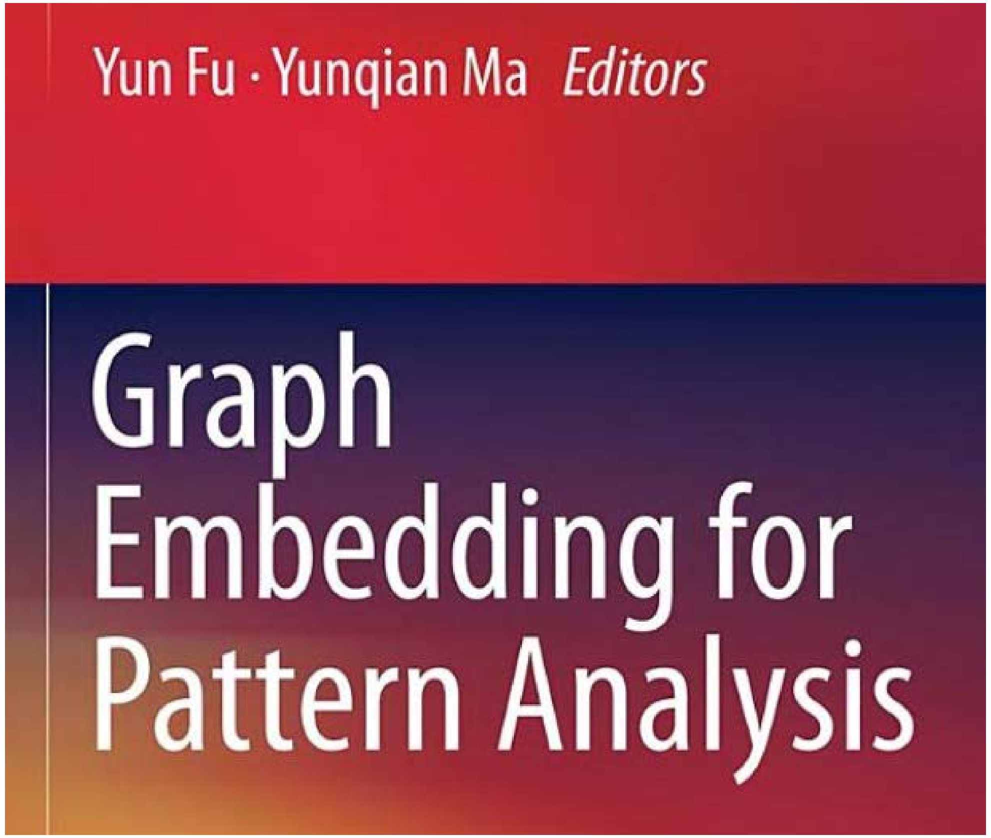
Vector-based representation of the Graph

Machine learning algorithms:

- Clustering, Classification
- Community Detection
- Anomaly Detection

Questions

- **What are the applications that can benefit from graph embedding?**
- **What is the benefit of the "intermediary" vector representation? Why not to use direct graph mining techniques?**
  - For example, label propagation for community detection?
- **What are available graph embedding algorithms?**
  - How do they compare with each other?
  - Where does DeepWalk stand in that category?
  - Is there anything better (in what regard)?


Yun Fu · Yunqian Ma Editors
Graph Embedding for Pattern Analysis

Why “DeepWalk” is so interesting?

- **DeepWalk algorithm maps Graph Embedding problem onto Word Embedding problem and takes advantage of:**
    - the recent advances in Natural Language Processing (NLP), specifically, in word embedding (Word2Vec, GloVe).

NLP: Natural Language Processing

VECTOR SPACE MODEL VS. WORD EMBEDDING MODEL

NLP (Understanding Language)

- **Discipline interested in creating algorithms to understand and process text**
- **Problems in this space include:**
  - Sentiment Analysis (finding out how people feel about something)
  - Topic Discovery (finding topics or summaries of texts)
  - Document Search (finding documents and parts of documents that match a certain criterion)

NLP Operates on Documents and Corpa

- **NLP and Text Analysis works by processing collections of Documents, which are just defined as collections of tokens**
  - Examples include:
    - Web Pages, Books, Paragraphs, Sentences, Tweets, etc.
- **Documents in a corpus are generally in a human-readable representation, and need to be converted into a machine-readable representation**
  - Traditional model:
    - Document vector
  - Linear Embedding

Document Vectors

- **Documents are represented as "bags of words"**
- **Represented as *vectors* when used computationally**
  - A vector is like an array of floating point
  - Has direction and magnitude
  - Each vector holds a place for *every word/term* in the collection
  - Therefore, most vectors are ***sparse*** (i.e., lots of zero's)

Vector Representation

- **Documents are represented as *vectors***
- **Position 1 corresponds to word 1, position 2 to word 2, position t to word t**
- **The weight of the word/term in the document is stored in each position**

$$D_i = w_{d_{i1}}, w_{d_{i2}}, \ldots, w_{d_{it}}$$

$$w = 0 \text{ if a term is absent, } 1 \text{ if present}$$

Document Vectors + Frequency

ID	nova	galaxy	heat	h'wood	film	role	diet	fur
A	10	5	3					
B								
C								
D								
E								
F								
G								
H								
I								

"Nova" occurs 10 times in text A
"Galaxy" occurs 5 times in text A
"Heat" occurs 3 times in text A
(Blank means 0 occurrences.)

Document Vectors + Frequency

ID	nova	galaxy	heat	h'wood	film	role	diet	fur
A								
B								
C								
D								
E								
F								
G								
H								
I				7	5		1	3

"Hollywood" occurs 7 times in text I
"Film" occurs 5 times in text I
"Diet" occurs 1 time in text I
"Fur" occurs 3 times in text I

Vector Space Model

ID	nova	galaxy	heat	h'wood	film	role	diet	fur
A	10	5	3					
B	5	10						
C				10	8	7		
D				9	10	5		
E							10	10
F							9	10
G	5		7			9		
H		6	10	2	8			
I				7	5		1	3

Issues with the Vector Space Model

1. **The Vector Space Model does not take the locality of words into play**
   - The sentence “I hated the cinematography” would still contribute to negative actor sentiment in the actor reviews
2. **Memory and computation of the vector space model scales as a function of both the number of documents and the number of unique words/terms**
   - To compute the cosine similarity, we need to perform $O(T)$ number of additions and multiplications
   - To store the vectors, we need a $O(D * T)$ matrix

Addressing Locality: Using Windows

Suppose that instead of using document vectors, we used word co-occurrence in windows

Example:

I loved Jane_Jones in loved movie. She is a good actress.

SW	Jane_Jones	loved	I	….
JJs	0	2	1	…
loved	2	0	1	…
I	1	1	0	…
…	…	…	…	…

Windows
I loved Jane_Jones
loved Jane_Jones in
Jane_Jones in loved
….

Notice that due to the sliding nature of the window, we need to make sure that the co-occurrence isn't overcounted

Approach with Word Co-occurrence

- **Strengths**
  - Incorporates word co-occurrence information

- **Weakness**
  - Generalized vector operations cannot be used on the data, only kernalized functions
  - Memory requirements are $O(T^2)$ (might be closer to $O(T)$ depending on the sparsity)
  - No communication occurs between sliding windows, meaning larger patterns could be missed

Defining a Proxy Function for Word Co-occurrence

- **Problem:**
  - Let $p(w_c|w_t)$ denote the probability of word w_c appearing with word w_t in a context window c centered on w_t (a normalized wc_{tc})

$$\underbrace{\text{I love Neural } \overset{w_t}{\boxed{\text{Networks}}} \text{ and Linear}}_{w_c} \text{ Embeddings!}$$

  - Can we find dense vectors representation for w_t and w_c, x_t and x_c, of size $d, d \ll |W|$ that proxies $p(w_t|w_c)$?

$$p(w_c|w_t) \approx \hat{p}(w_c|w_t) = n(v_t^T v_c)$$

where $v_t^T v_c$ is the dot product, and $n(x)$ normalizes $v_t^T v_c$

- Normalizing functions include the sigmoid and **softmax**

Finding the Values for x_i and x_j

- **Let $x_i = \{\beta_1^i, \beta_2^i, \dots, \beta_d^j\}$ and $x_j = \left\{\beta_1^j, \beta_2^j, \dots, \beta_d^j\right\}$**
  - The problem of finding x_i and x_j can then be expressed as the non-linear regression problem:

$$\hat{p}(w_c|w_t) = n(\beta_1^i\beta_1^j + \beta_2^i\beta_2^j + \cdots + \beta_d^i\beta_d^j)$$

  - This type of problem is usually solved by **Neural Networks**
- **This process, which finds parameters to a transformed linear operation to approximate values of a general relationship function, is called Linear Embedding**

Why use Embedding Vectors?

- **Strengths:**
  - Memory foot-print is small. Embedding Vectors are small dense vectors, with around 600 or fewer dimensions
  - The similarity between vectors has a semantic meaning (the cosine similarity was explicitly trained to represent element co-occurrence)
- **Weaknesses:**
  - Requires a large training set to produce good, non-noisy results
  - Embeddings are generally **static**, and cannot be easily iteratively updated. However, note that training an embedding is usually very quick (minutes)

Graph Embedding

RANDOM WALKS

Reminder: What is a Graph?

- **Graphs are mathematical objects used to encode relationships between elements in a collection**
- **A graph $G(V, E)$ is:**
  - A mathematical object defined by a set of nodes or vertices V and a set of edges E
  - Nodes represent the objects under consideration
  - Edges connect two nodes, and denote a relationship
    - Edges can be directed or undirected

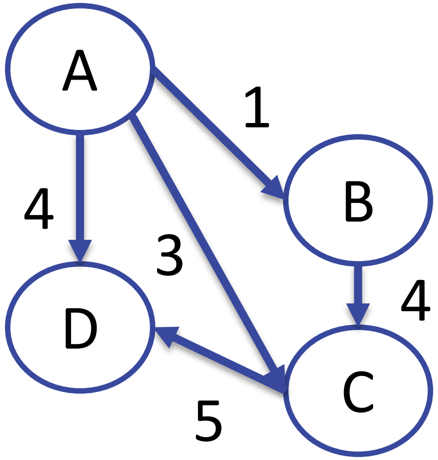
Graph Adjacency Matrix

- An **adjacency matrix**, $A^{|V|\times|V|}$ of the graph G is a matrix where:

$$a_{i,j} = \begin{cases} 1, & e_{i,j} \in E \\ 0, & if\ else \end{cases}$$

- Or in the case of a **weighted** graph

$$a_{i,j} = \begin{cases} w_{i,j}, & e_{i,j} \in E \\ 0, & if\ else \end{cases}$$



Nodes	A	B	C	D
A	0	1	3	4
B	0	0	5	0
C	0	0	0	5
D	0	0	0	0

The Adjacency Matrix is Similar to the Bag-of-words

Bag-of-Words

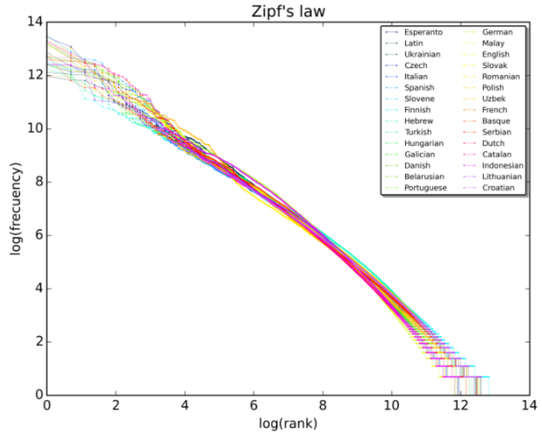
ID	nova	galaxy	heat	h'wood	film	role	diet	fur
A	10	5	3					
B	5	10						
C				10	8	7		
D				9	10	5		
E							10	10
F							9	10
G	5		7			9		
H		6	10	2	8			
I				7	5		1	3

Adjacency Matrix

Nodes	A	B	C	D
A	0	1	3	4
B	0	0	5	0
C	0	0	0	5
D	0	0	0	0

Stronger Evidence: Power-law distributions

- **The distribution of the frequency of words follows a power-law distribution (Zipf's Law)**
  - $d(x) \approx x^{-1.5}$



- **Graphs that describe natural systems are often scale-free**
  - The node degree (i.e., the number of adjacent nodes) distribution follows a power law,
  - $d(x) = x^{-\alpha}, 2 < \alpha < 3$

Both Are Power Laws!

An Issue with Embedding: What is a "context" for a graph?

- **Words have a natural context with sentences. Surrounding words give context to the target**

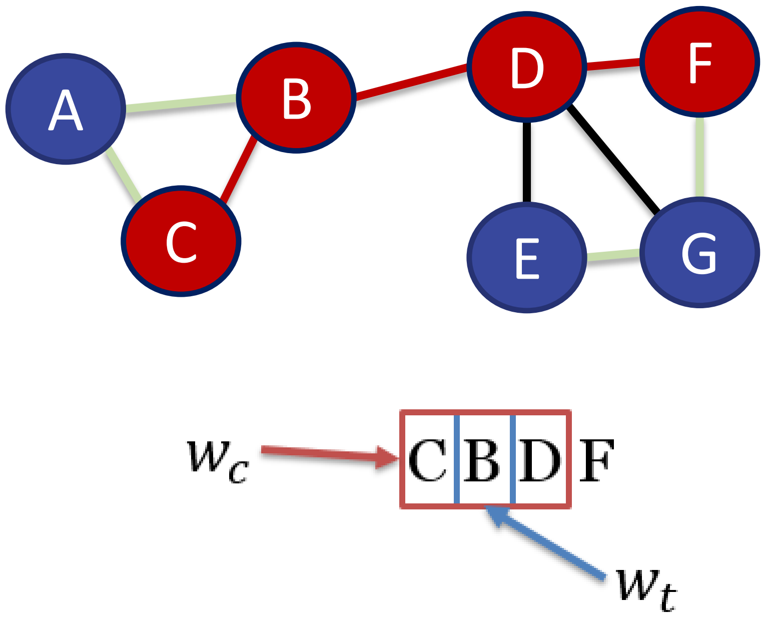
I love Neural Networks and Linear Embeddings!

- **However, the columns and rows of an adjacency matrix are arbitrary, and cannot be sampled for context**

Nodes	A	B	C	D
A	0	1	3	4
B	0	0	5	0
C	0	0	0	5
D	0	0	0	0

One Approach to Sample for Context: Random Walks

- **Suppose two nodes, u and v are contextually similar**
  - If we start at u and perform a random walk of length l from u, we can form a "sentence" from the walk
  - Neighbors should be in the context of other neighbors, meaning the embedding should pair close neighbors

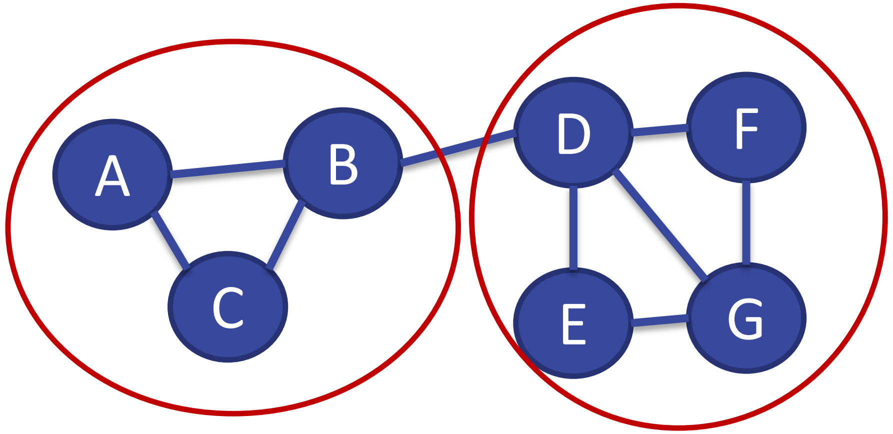


Graph Embedding

COMMUNITY DETECTION AND DEEPWALK

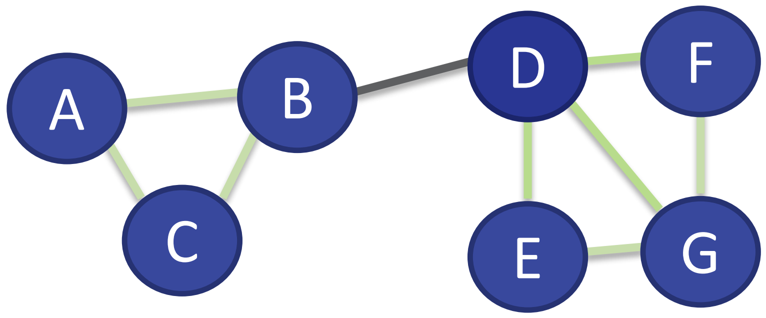
Graph Data Mining Problem: Community Detection

- **A common problem in graph analysis is the problem of community detection**
  - Given a graph $G(V,E)$, can we divide the nodes into a collection of **subgraphs** S, where the nodes inside the graphs are **more connected inside than they are to the outside graph**



One Approach to Community Detection: Random Walks

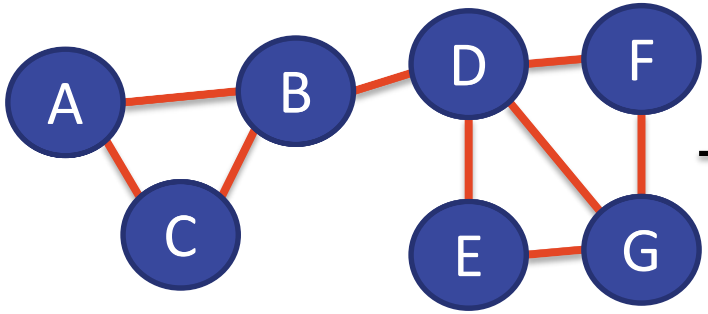
- **Suppose two nodes, u and v belong to the same community c**
  - If we start at u and perform a random walk of length l from u, we should find v in a lot of the random walks
    - The elements in the community are more connected to the elements in the community.
    - Therefore, they should be selected at higher rates



Nodes around D	Prop of selection
F	.25
G	.25
E	.25
B	.25

DeepWalk: Computing Graph Embedding for Community Detection

- Sample a set of random walks starting at every node in the graph
- Treating these random walks as sentences, pass the new “corpus” into a Neural Network (Word2Vec)
- Cluster the new “word” vectors using k-medoid clustering. Every cluster is a community in the graph



A B C
B D F
G E D
C B A
D F G
E D G
E G F

Dim	1	2	3
A	.4	.2	.6
B	.5	.2	.9
C	.5	.5	.5
D	.2	.6	.2
….	….	….	….

Using graphs to combine embeddings

HETEROGENEOUS DATA FUSION

Heterogeneous Data

- **Currently, we have assumed data belonged to the same modality**
  - Modality: a class of data with the same inherit structure and behavior, a “type” of data
  - Examples:
    - Text
    - Images
    - Video
    - Sound
- **If the data of interest contains data from multiple modalities, it is called a heterogeneous data source**

Heterogeneous Data Fusion

- **The process of analyzing and combining information from multiple modalities to answer a series of questions is called Heterogeneous Data Fusion**
  - For example, given a text review and pictures of a restaurant, can we determine what rating and tags will be assigned to a restaurant by patrons?

….natural lighting and a light breeze……. +

=

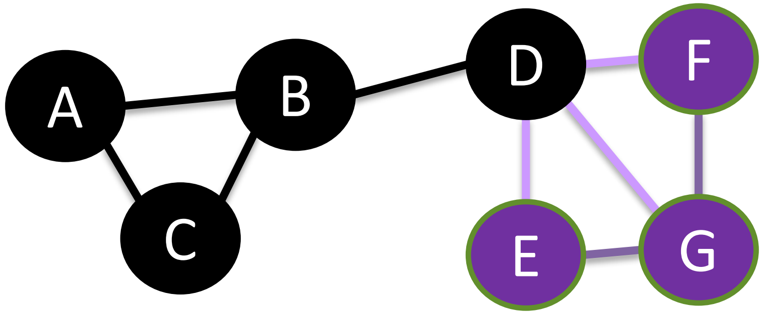
Tags: Outside
Rating: 4

The Process of Heterogeneous Data Fusion

1. **Define the relationships between different elements of the data**
   - Which elements should we consider "close" together
2. **Define, for each modality, a neural network representing the embedding**
   - For each modality, we need to define an embedding architecture to generate an embedding vector x_k for each element in that modality
     - Example: Text: skip-Gram Model and Image: Convolutional Neural Network (CNN)
3. **Using the embeddings from each architecture, learn a new embedding vector z in space r that allows us to directly compare different embeddings**
4. **Define an error function that will allow us to learn all the embeddings in tandem**

Define the relations between different elements of the data

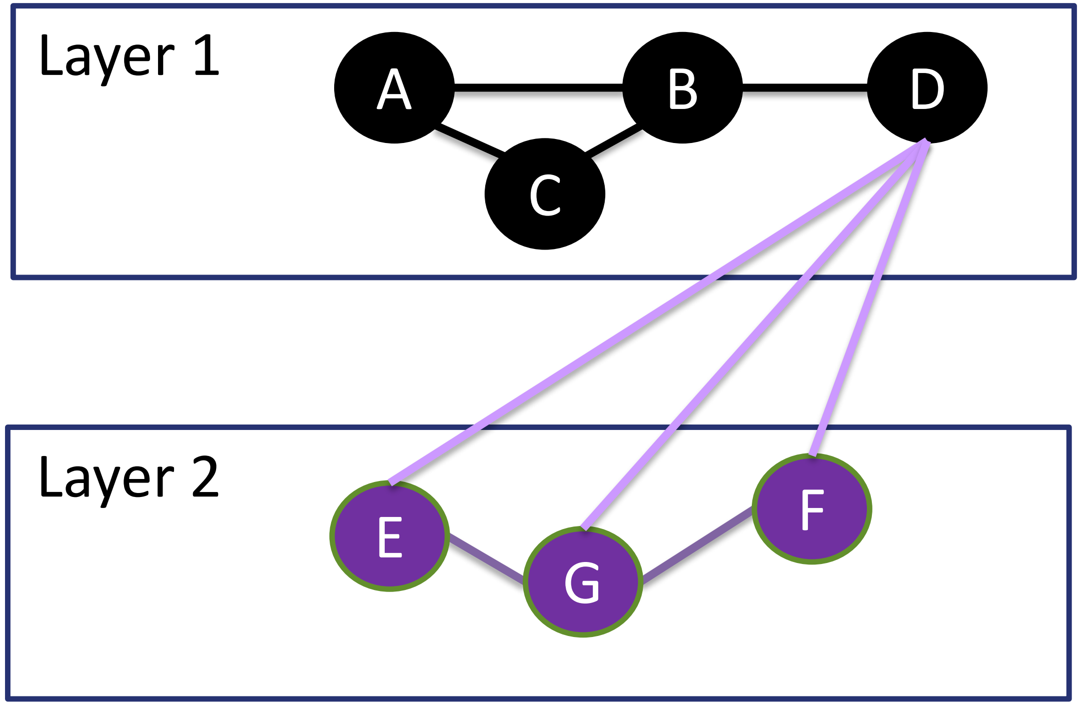
- **Let G=(V,E) denote a graph of relationships, E, between different objects in V**



- **Each node belongs to a class $\mathcal{O}$**
  - $\mathcal{O} \in \{black, purple\}$
- **Each edge belongs to a class $\mathcal{R}$**
  - $\mathcal{R} \in \{bb, pp, bp\}$

Define the **relations** between different elements of the data

- **Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a multi-graph with two layers**
  - $\mathcal{V} = \{V_1, V_2\}$
  - $\mathcal{E} = \{E_1, E_2, E_{1,2}\}$



Heterogeneous Data Fusion Summary

- **To answer more complex questions, sometimes we need to project multiple embeddings describing different types of data into the same space**
- **The process of projection is the same as previous embedding methods**
  1. Define an initial representation for each type of data
  2. Create a new embedding layer that projects each representation into the same space
  3. Create a new error function that is the sum of the sigmoid functions of every combination of elements in the data, along with a complexity penalty
  4. Train using Neural Network Training Procedures

Graph Embedding

EXTRA SLIDES

Recent publications in the field of Graph Embedding

- DeepWalk: Online Learning of Social Representations
- Efficient Estimation of Word Representations ("Word2Vec")
- Comprehend DeepWalk as Matrix Factorization
- LINE: Large-scale Information Network Embedding
- Glove: Global Vectors for Word Representation
- Distributed Large-scale Natural Graph Factorization

LINE: Large-scale Information Network Embedding

- **Main ideas: Algorithm utilizes a "carefully" designed objective function that preserves both first-order and second-order proximities**
- **Uses negative sampling – not hierarchical softmax**
- **Uses an edge-sampling algorithm for optimizing the objective.**
- **Code: https://github.com/tangjianpku/LINE**

LINE on DeepWalk

- "… DeepWalk, which deploys a truncated random walk for social network embedding. Although empirically effective, the DeepWalk does not provide a clear objective that articulates what network properties are preserved. Intuitively, DeepWalk expects nodes with higher second-order proximity yield similar low-dimensional representations, while LINE preserves both first-order and second-order proximities.
- DeepWalk uses random walks to expand the neighborhood of a vertex, which is analogous to depth-first search (DFS). LINE uses breadth-first search (BFS) strategy which is more reasonable approach to the second-order proximity.
- DeepWalk only applies to unweighted…

First-order proximity in a network is …

- … the local pairwise proximity between two vertices.
- For each pair of vertices linked by an edge (u, v) , the weight on that edge, w_{uv} , indicates the first order proximity between u and v.
- If no edge is observed between u and v, their first-order proximity is 0.

Second-order proximity

- **Between a pair of vertices (u, v) in a network is the similarity between their neighborhood structures.**
- **Let $p_u = (w_{u,1}, \dots, w_{u,|V|})$ denote the first order proximity of u with all the other vertices, then the second-order proximity between u and v is determined by the similarity between p_u and p_v**
- **If no vertex is linked from/to both u and v , the second-order proximity between u and v is 0.**

Large-scale Information Network Embedding

- **Given a large network $G = (V, E)$, the Large-scale Information Network Embedding aims to represent each vertex $v \in V$ into a low-dimensional space R^d**
  - i.e. learning a function $f_G: V \to R^d$, where $d \ll |V|$
  - In the space R^d, both the first-order proximity and second-order proximity between the vertices are preserved.

LINE with first order proximity

- **For each undirected edge (i, j) the joint probability between vertex v_i and v_j**
  - $p_1(v_i, v_j) = \frac{1}{1+\exp(-\boldsymbol{u}_i^T \cdot \boldsymbol{u}_j)}$
  - where $\boldsymbol{u}_i \in R^d$ is the low-dimensional vector representation of vertex v_i
- **Defines a distribution $p(\cdot,\cdot)$ over the space $V \times V$**
  - its empirical probability can be defined as $\hat{p}(i, j) = \frac{w_{ij}}{W}$,
  - where $W = \sum_{(i,j)\in E} w_{ij}$

Testing LINE: Word Analogy task

- This task is introduced by Mikolov et al. [12]. Given a word pair (a; b) and a word c, the task aims to nd a word d, such that the relation between c and d is similar to the relation between a and b, or denoted as: a : b ! c :?.

Table 2: Results of word analogy on WIKIPEDIA data.

Algorithm	Semantic (%)	Syntactic (%)	Overall (%)	Running time
GF	61.38	44.08	51.93	2.96h
DeepWalk	50.79	37.70	43.65	16.64h
SkipGram	69.14	57.94	63.02	2.82h
LINE-SGD(1st)	9.72	7.48	8.50	3.83h
LINE-SGD(2nd)	20.42	9.56	14.49	3.94h
LINE(1st)	58.08	49.42	53.35	2.44h
LINE(2nd)	**73.79**	**59.72**	**66.10**	2.55h

Testing LINE: page classification task

Table 3: Results of Wikipedia page classification on WIKIPEDIA data set.

Metric	Algorithm	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1	GF	79.63	80.51	80.94	81.18	81.38	81.54	81.63	81.71	81.78
	DeepWalk	78.89	79.92	80.41	80.69	80.92	81.08	81.21	81.35	81.42
	SkipGram	79.84	80.82	81.28	81.57	81.71	81.87	81.98	82.05	82.09
	LINE-SGD(1st)	76.03	77.05	77.57	77.85	78.08	78.25	78.39	78.44	78.49
	LINE-SGD(2nd)	74.68	76.53	77.54	78.18	78.63	78.96	79.19	79.40	79.57
	LINE(1st)	79.67	80.55	80.94	81.24	81.40	81.52	81.61	81.69	81.67
	LINE(2nd)	79.93	80.90	81.31	81.63	81.80	81.91	82.00	82.11	82.17
	LINE(1st+2nd)	**81.04****	**82.08****	**82.58****	**82.93****	**83.16****	**83.37****	**83.52****	**83.63****	**83.74****
Macro-F1	GF	79.49	80.39	80.82	81.08	81.26	81.40	81.52	81.61	81.68
	DeepWalk	78.78	79.78	80.30	80.56	80.82	80.97	81.11	81.24	81.32
	SkipGram	79.74	80.71	81.15	81.46	81.63	81.78	81.88	81.98	82.01
	LINE-SGD(1st)	75.85	76.90	77.40	77.71	77.94	78.12	78.24	78.29	78.36
	LINE-SGD(2nd)	74.70	76.45	77.43	78.09	78.53	78.83	79.08	79.29	79.46
	LINE(1st)	79.54	80.44	80.82	81.13	81.29	81.43	81.51	81.60	81.59
	LINE(2nd)	79.82	80.81	81.22	81.52	81.71	81.82	81.92	82.00	82.07
	LINE(1st+2nd)	**80.94****	**81.99****	**82.49****	**82.83****	**83.07****	**83.29****	**83.42****	**83.55****	**83.66****

Significantly outperforms GF at the: ** 0.01 and * 0.05 level, paired t-test.

Graph embedding methods cited in "LINE"

- **Multidimensional scaling (MDS)**
- **IsoMap**
- **Laplacian eigenmap**
  - These algorithms usually rely on solving the leading eigenvectors of the affinity matrices, the complexity of which is at least quadratic to the number of nodes, making them inefficient to handle large-scale networks.
- **NEW and powerful: Graph Factorization**
  - It finds the low-dimensional embedding of a large graph through matrix factorization, which is optimized using stochastic gradient descent. This is possible because a graph can be represented as an affinity matrix. However, the objective of matrix factorization is not designed for networks, therefore does not necessarily preserve the global network structure.

Multidimensional scaling: MDS

Multidimensional scaling (**MDS**) is a means of visualizing the level of similarity of individual cases of a dataset. It refers to a set of related ordination techniques used in information visualization, in particular to display the information contained in a distance matrix. An MDS algorithm aims to place each object in *N*-dimensional space such that the between-object distances are preserved as well as possible. Each object is then assigned coordinates in each of the *N* dimensions. The number of dimensions of an MDS plot *N* can exceed 2 and is specified a priori. Choosing *N*=2 optimizes the object locations for a two-dimensional scatterplot.[1]

Isomap

Isomap

From Wikipedia, the free encyclopedia

Isomap is a Nonlinear dimensionality reduction method. It is one of several widely used low-dimensional embedding methods.[1] Isomap is used for computing a quasi-isometric, low-dimensional embedding of a set of high-dimensional data points. The algorithm provides a simple method for estimating the intrinsic geometry of a data manifold based on a rough estimate of each data point's neighbors on the manifold. Isomap is highly efficient and generally applicable to a broad range of data sources and dimensionalities.

Contents [show]

Introduction [edit]

Isomap is one representative of isometric mapping methods, and extends metric multidimensional scaling (MDS) by incorporating the geodesic distances imposed by a weighted graph. To be specific, the classical scaling of metric MDS performs low-dimensional embedding based on the pairwise distance between data points, which is generally measured using straight-line Euclidean distance. Isomap is distinguished by its use of the geodesic distance induced by a neighborhood graph embedded in the classical scaling. This is done to incorporate manifold structure in the resulting embedding. Isomap defines the geodesic distance to be the sum of edge weights along the shortest path between two nodes (computed using Dijkstra's algorithm, for example). The top n eigenvectors of the geodesic distance matrix, represent the coordinates in the new n-dimensional Euclidean space.

Graph embedding methods cited in "DeepWalk"

- **SpectralClustering**
  - SpectralClustering [41]: This method generates a representation in $\mathbb{R}^d$ from the d-smallest eigenvectors of $\widetilde{\mathcal{L}}$, the normalized graph Laplacian of G. Utilizing the eigenvectors of $\widetilde{\mathcal{L}}$ implicitly assumes that graph cuts will be useful for classification.

- **Modularity**
  - Modularity [39]: This method generates a representation in $\mathbb{R}^d$ from the top-d eigenvectors of B, the Modularity matrix of G. The eigenvectors of B encode information about modular graph partitions of G [35]. Using them as features assumes that modular graph partitions will be useful for classification.

- **EdgeCluster**
  - EdgeCluster [40]: This method uses k-means clustering to cluster the adjacency matrix of G. Its has been shown to perform comparably to the Modularity method, with the added advantage of scaling to graphs which are too large for spectral decomposition.

- **wvRN**
  - wvRN [25]: The weighted-vote Relational Neighbor is a relational classifier. Given the neighborhood $\mathcal{N}_i$ of vertex v_i, wvRN estimates $\Pr(y_i|\mathcal{N}_i)$ with the (appropriately normalized) weighted mean of its neighbors (i.e $\Pr(y_i|\mathcal{N}_i) = \frac{1}{Z}\sum_{v_j \in \mathcal{N}_i} w_{ij} \Pr(y_j \mid \mathcal{N}_j)$). It has shown surprisingly good performance in real networks, and has been advocated as a sensible relational classification baseline [26].