Recent Advances on Graph Analytics and Its Applications in Healthcare

KDD 2020 Tutorial
August 23, morning
Fei Wang, Peng Cui, Jian Pei, Yangqiu Song, Chengxi Zang,
Network Embedding and Graph Neural Networks

Peng Cui
Tsinghua University
Many healthcare problems can be modeled as graph problems.

Drug retargeting

Adverse drug reaction

http://www.cytoscape.org/

Networks are not *learning-friendly*

$G = (V, E)$

- **Inapplicability of ML methods**
- **Links** → **Topology**

Pipeline for network analysis:

1. Network Data
2. Feature Extraction
3. Pattern Discovery
4. Network Applications
Learning from networks

Network Embedding

GCN
Network Embedding

\[ G = (V, E) \]

- Easy to parallel
- Can apply classical ML methods
The goal of network embedding

**Goal** Support network inference in vector space

- Reflect network structure
- Maintain network properties

Transform network nodes into vectors that are fit for off-the-shelf machine learning models.
Graph Neural Networks

- Basic idea: recursive definition of states

\[ s_i = \sum_{j \in \mathcal{N}(i)} F \left( s_i, s_j, F_i^V, F_j^V, F_{i,j}^E \right) \]

- A simple example: PageRank

Graph Convolutional Networks (GCN)

- Main idea: pass messages between pairs of nodes & agglomerate

\[ H^{l+1} = \rho \left( \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^l \Theta^l \right) \]

- Stacking multiple layers like standard CNNs:
  - State-of-the-art results on node classification

---

Network Embedding and GCN

![Diagram showing network embedding and GCN process]

- **Input**: Graph and Feature
- **Model**: Network Embedding and GCN
- **Output**: Embedding and Task results

**Key Points**:
- **Topology to Vector**
- **Fusion of Topology and Features**
- **Unsupervised v.s. (Semi-)Supervised**
Learning from networks

Network Embedding

GCN
The intrinsic problems NE is solving

Reducing representation dimensionality while preserving necessary topological structures and properties.

- Nodes & Links
- Node Neighborhood
- Pair-wise Proximity
- Community
- Hyper Edges
- Global Structure

- Non-transitivity
- Asymmetric Transitivity
- Uncertainty
- Dynamic
- Heterogeneity
- Interpretability
Preserving Arbitrary-Order Proximity

• Shifting across different orders/weights:
  - Preserving arbitrary-order proximity
  - Low marginal cost
  - Accurate and efficient

Preserving Arbitrary-Order Proximity

- High-order proximity: a polynomial function of the adjacency matrix
  \[ S = f(A) = w_1 A^1 + w_2 A^2 + \cdots + w_q A^q \]
  - \( q \): order; \( w_1 \ldots w_q \): weights, assuming to be non-negative
  - \( A \): could be replaced by other variations (such as the Laplacian matrix)

- Objective function: matrix factorization
  \[ \min_{U^*, V^*} \| S - U^* V^*^T \|_F^2 \]
  - \( U^*, V^* \in \mathbb{R}^{N \times d} \): left/right embedding vectors
  - \( d \): dimensionality of the space

- Optimal solution: Singular Value Decomposition (SVD)
  - \([U, \Sigma, V]\): top-d SVD results
    \[ U^* = U \sqrt{\Sigma}, \quad V^* = V \sqrt{\Sigma} \]
Preserving Arbitrary-Order Proximity

- Eigen-decomposition reweighting

**Theorem 4.2 (Eigen-Decomposition Reweighting).** If $[\lambda, x]$ is an eigen-pair of $A$, then $[\mathcal{F}(\lambda), x]$ is an eigen-pair of $S = \mathcal{F}(A)$.

Experimental Results

- Link Prediction

Hyper-network embedding

• A hyper-network is a network in which an edge can include any number of nodes
Hyper-edges are often **indecomposable**

Structural Deep Network for Hyper-network

Experiment: link prediction

<table>
<thead>
<tr>
<th>methods</th>
<th>GPS</th>
<th>MovieLens</th>
<th>drug</th>
<th>wordnet</th>
</tr>
</thead>
<tbody>
<tr>
<td>DHNE</td>
<td>0.9166</td>
<td>0.8676</td>
<td>0.9254</td>
<td>0.8268</td>
</tr>
<tr>
<td>deepwalk</td>
<td>0.6593</td>
<td>0.7151</td>
<td>0.5822</td>
<td>0.5952</td>
</tr>
<tr>
<td>line</td>
<td>0.7795</td>
<td>0.7170</td>
<td>0.7057</td>
<td>0.6819</td>
</tr>
<tr>
<td>node2vec</td>
<td>0.5835</td>
<td>0.8211</td>
<td>0.6573</td>
<td>0.8003</td>
</tr>
<tr>
<td>SHE</td>
<td>0.8687</td>
<td>0.7450</td>
<td>0.5899</td>
<td>0.5426</td>
</tr>
<tr>
<td>min</td>
<td>0.5715</td>
<td>0.6307</td>
<td>0.5493</td>
<td>0.5542</td>
</tr>
<tr>
<td>deepwalk</td>
<td>0.7219</td>
<td>0.6265</td>
<td>0.7651</td>
<td>0.6225</td>
</tr>
<tr>
<td>line</td>
<td>0.5869</td>
<td>0.7675</td>
<td>0.6546</td>
<td>0.7985</td>
</tr>
<tr>
<td>node2vec</td>
<td>0.8078</td>
<td>0.8012</td>
<td>0.6508</td>
<td>0.5507</td>
</tr>
<tr>
<td>SHE</td>
<td>0.8646</td>
<td>0.7201</td>
<td>0.6470</td>
<td>0.6516</td>
</tr>
<tr>
<td>tensor</td>
<td>0.8355</td>
<td>0.7740</td>
<td>0.8191</td>
<td>0.6364</td>
</tr>
<tr>
<td>HEBE</td>
<td>0.8355</td>
<td>0.7740</td>
<td>0.8191</td>
<td>0.6364</td>
</tr>
</tbody>
</table>

The overall performance

Performance on networks of different sparsity

Learning from networks

Network Embedding

GCN
The intrinsic problem GCN is solving

Fusing topology and features in the way of smoothing features with the assistance of topology.

\[ \mathbf{H}^{l+1} = \rho \left( \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{H}^l \Theta^l \right) \]
Robust GCN

- Adversarial attacks
  - small perturbations in graph structures and node attributes
  - great challenges for applying GCNs to node classification
Gaussian Based hidden representations: Variance terms absorb the effects of adversarial attacks

Attention mechanism: Remedy the propagation of adversarial attacks

Sampling process: Explicitly considers mathematical relevance between means and variances

Robust GCN

- Node Classification on Clean Datasets

<table>
<thead>
<tr>
<th></th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>81.5</td>
<td>70.9</td>
<td>79.0</td>
</tr>
<tr>
<td>GAT</td>
<td>83.0</td>
<td>72.5</td>
<td>79.0</td>
</tr>
<tr>
<td>RGCN</td>
<td>83.1</td>
<td>71.3</td>
<td>79.2</td>
</tr>
</tbody>
</table>

- Against Non-targeted Adversarial Attacks

Figure 2: Results of different methods when adopting Random Attack as the attack method.

Disentangled GCN

- A real-world graph is typically formed due to *many* latent factors.

- Existing GNNs/GCNs:
  - A holistic approach, that takes in the *whole* neighborhood to produce a *single* node representation.

- We suggest:
  - To disentangle the latent factors.
    - (By segmenting the heterogeneous parts, and learning multiple factor-specific representations for a node.)
  - Robustness (e.g., not overreact to an irrelevant factor) & Interpretability.
We present DisenGCN, the disentangled graph convolutional network.

- DisenConv, a disentangled multichannel convolutional layer (figure below).
- Each channel convolutes features related with a single latent factor.

Disentangled GCN

(a) GCN.  
(b) DisenGCN (this work).

Some interesting questions for GCN...
What if the problem is topology-driven?

- Since GCN is filtering features, it is inevitably feature-driven
  - Structure only provides auxiliary information (e.g. for filtering/smoothing)
- When feature plays the key role, GNN performs good …
- How about the contrary?
- Synthesis data: stochastic block model + random features

<table>
<thead>
<tr>
<th>Method</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>10.0</td>
</tr>
<tr>
<td>GCN</td>
<td>18.3 ± 1.1</td>
</tr>
<tr>
<td>DeepWalk</td>
<td>99.0 ± 0.1</td>
</tr>
</tbody>
</table>
Does GCN fuse feature and topology optimally?

**Fusion Capability of GCNs**

- Ideal Solution: extract the most correlated information for task

---

**Case 1**
- Random topology
- Correlated Features

MLP(100%) > GCN(75.2%)

**Case 2**
- Correlated Topology
- Random Features

DeepWalk(100%) > GCN(87%)

Rethinking: Is GCN truly a Deep Learning method?

- Recall GNN formulation:
  \[ H^{(k+1)} = \sigma(SH^{(k)}W^{(k)}), S = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} \]

- How about removing the non-linear component:
  \[ H^{(k+1)} = SH^{(k)}W^{(k)} \]

- Stacking multiple layers and add softmax classification:
  \[ \hat{Y} = softmax(H^{(K)}) \]
  \[ = softmax(SS ... SH^{(0)}W^{(0)}W^{(1)} ... W^{(K-1)}) \]
  \[ = softmax(S^KH^{(0)}W) \]

Rethinking: Is GCN truly a Deep Learning method?

This simplified GNN (SGC) shows remarkable results:

<table>
<thead>
<tr>
<th></th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>81.4 ± 0.4</td>
<td>70.9 ± 0.5</td>
<td>79.0 ± 0.4</td>
</tr>
<tr>
<td>GAT</td>
<td>83.3 ± 0.7</td>
<td>72.6 ± 0.6</td>
<td>78.5 ± 0.3</td>
</tr>
<tr>
<td>FastGCN</td>
<td>79.8 ± 0.3</td>
<td>68.8 ± 0.6</td>
<td>77.4 ± 0.3</td>
</tr>
<tr>
<td>GIN</td>
<td>77.6 ± 1.1</td>
<td>66.1 ± 0.9</td>
<td>77.0 ± 1.2</td>
</tr>
<tr>
<td>LNet</td>
<td>80.2 ± 3.0†</td>
<td>67.3 ± 0.5</td>
<td>78.3 ± 0.6†</td>
</tr>
<tr>
<td>AdaLNet</td>
<td>81.9 ± 1.9†</td>
<td>70.6 ± 0.8†</td>
<td>77.8 ± 0.7†</td>
</tr>
<tr>
<td>DGI</td>
<td>82.5 ± 0.7</td>
<td>71.6 ± 0.7</td>
<td>78.4 ± 0.7</td>
</tr>
<tr>
<td>SGC</td>
<td>81.0 ± 0.0</td>
<td>71.9 ± 0.1</td>
<td>78.9 ± 0.0</td>
</tr>
</tbody>
</table>

Text Classification

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Test Acc. ↑</th>
<th>Time (seconds) ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>20NG</td>
<td>GCN</td>
<td>87.9 ± 0.2</td>
<td>1205.1 ± 144.5</td>
</tr>
<tr>
<td></td>
<td>SGC</td>
<td>88.5 ± 0.1</td>
<td>19.06 ± 0.15</td>
</tr>
<tr>
<td>R8</td>
<td>GCN</td>
<td>97.0 ± 0.2</td>
<td>129.6 ± 9.9</td>
</tr>
<tr>
<td></td>
<td>SGC</td>
<td>97.2 ± 0.1</td>
<td>1.90 ± 0.03</td>
</tr>
<tr>
<td>R52</td>
<td>GCN</td>
<td>93.8 ± 0.2</td>
<td>245.0 ± 13.0</td>
</tr>
<tr>
<td></td>
<td>SGC</td>
<td>94.0 ± 0.2</td>
<td>3.01 ± 0.01</td>
</tr>
<tr>
<td>Ohsumed</td>
<td>GCN</td>
<td>68.2 ± 0.4</td>
<td>252.4 ± 14.7</td>
</tr>
<tr>
<td></td>
<td>SGC</td>
<td>68.5 ± 0.3</td>
<td>3.02 ± 0.02</td>
</tr>
<tr>
<td>MR</td>
<td>GCN</td>
<td>76.3 ± 0.3</td>
<td>16.1 ± 0.4</td>
</tr>
<tr>
<td></td>
<td>SGC</td>
<td>75.9 ± 0.3</td>
<td>4.00 ± 0.04</td>
</tr>
</tbody>
</table>

Network Embedding v.s. GCN

There is no better one, but there is more proper one.
Summaries and Conclusions

- Unsupervised v.s. (Semi-)Supervised
- Topology-driven v.s. Feature-driven
- For different healthcare tasks, there is no best one, but there is more proper one.
A Survey on Network Embedding

Deep Learning on Graphs: A Survey

Ziwei Zhang, Peng Cui and Wenwu Zhu

Abstract—Deep learning has been shown successful in a number of domains, ranging from acoustics, images to natural language processing. However, applying deep learning to the ubiquitous graph data is non-trivial because of the unique characteristics of graphs. Recently, a significant amount of research efforts have been devoted to this area, greatly advancing graph analyzing techniques. In this survey, we comprehensively review different kinds of deep learning methods applied to graphs. We divide existing methods into three main categories: semi-supervised methods including Graph Neural Networks and Graph Convolutional Networks, unsupervised methods including Graph Autoencoders, and recent advancements including Graph Recurrent Neural Networks and Graph Reinforcement Learning. We then provide a comprehensive overview of these methods in a systematic manner following their history of developments. We also analyze the differences of these methods and how to composite different architectures. Finally, we briefly outline their applications and discuss potential future directions.

Index Terms—Graph Data, Deep Learning, Graph Neural Network, Graph Convolutional Network, Graph Autoencoder.

1 INTRODUCTION

In the last decade, deep learning has been a “crown jewel” in artificial intelligence and machine learning [1], showing superior performance in acoustics [2], images [3] and natural language processing [4]. The expressive power of deep learning to extract complex patterns underlying data has been well recognized. On the other hand, graphs are ubiquitous in the real world, repre-
Thanks!

Peng Cui
cuip@tsinghua.edu.cn
http://pengcui.thumedialab.com