Motivations

Feature Selection

Network Embedding

Attributed Network Embedding

Graph Neural Networks
The problems of network representation

\[ G = (V, E) \]

- Iterative & Combinatorial Complexity
- Coupling
- Parallelizability
- Computability
The problems of network representation

G = (V, E)

Pipeline for network analysis

Network Data

Feature Extraction

Pattern Discovery

Network Applications

Inapplicability of ML methods

Links → Topology

Learnability
Revisit network representation

\[ G = (V, E) \]

- Easy to parallel
- Can apply classical ML methods

Vector Space

\[ G = (V) \]
The ultimate goal

Network Inference
- Node importance
- Community detection
- Network distance
- Link prediction
- Node classification
- Network evolution
- ...

in Vector Space
The information encoded in networks

Topological Information

Semantic Information

Nodes Attributes

Topology Vectorization is the key problem.
**Goal**  Support network inference in vector space

Reflect network structure  
Maintain network properties

Network Embedding

Transitivity
Outline

• Structure-preserved network embedding
• Property-preserved network embedding
• Dynamic network embedding
• Robustness, Explainability and Applicability
• Network embedding for biomedical applications
Outline

- Structure-preserved network embedding
- Property-preserved network embedding
- Dynamic network embedding
- Robustness, Explainability and Applicability
- Network embedding for biomedical applications
Network Structures

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

Reconstruct the original network

Matrix Factorization

Reconstruct all the links? May cause overfitting. The network inference ability is seriously limited.
Network Structures

Nodes & Links

Pair-wise Proximity

Community Structures

Hyper Edges

Global Structure
High-Order Proximity

• Capturing the underlying structure of networks

• Advantages:
  • Solve the sparsity problem of network connections
  • Measure indirect relationship between nodes
Deepwalk

- Exploit truncated random walk to define neighborhood of a node.

Random Walks on Graph

- $V_{26} - V_{25} - V_{32} - V_3 - V_{10} \ldots$
- $V_5 - V_7 - V_{17} - V_6 - V_{11} \ldots$
- $V_{31} - V_{33} - V_{21} - V_{33} - V_{15}$

LINE

LINE with First-order Proximity:
local pairwise

\[ O_1 = - \sum_{(i,j) \in E} w_{i,j} \log p_1(v_i, v_j) \]

LINE with Second-order Proximity:
neighborhood structures

\[ O_2 = \sum_{i \in V} \lambda_i d(\hat{p}_2(\cdot | v_i), p_2(\cdot | v_i)) \]

SDNE – Structural Deep Network Embedding

Unsupervised Autoencoder (preserve second-order proximity)

Unsupervised Autoencoder (preserve second-order proximity)

GraRep: Learning Graph Representations with Global Structural Information.

1-step

2-step

3-step

4-step

Do not distinguish 1-step and 2-step.

capturing different k-step information

maintaining different k-step information separately

What is the right order?

- Different networks/tasks require different high-order proximities
  - E.g., multi-scale classification (Bryan Perozzi, et al, 2017)
  - E.g., networks with different scales and sparsity
  - Proximities of different orders can also be arbitrarily weighted
    - E.g., equal weights, exponentially decayed weights (Katz)
What is the *right* order?

- Existing methods can only preserve one fixed high-order proximity
  - Different high-order proximities are calculated separately

- \( \text{Proximity1} \rightarrow \text{Embedding1} \)
- \( \text{Proximity2} \rightarrow \text{Embedding2} \)
- \( \text{Proximity3} \rightarrow \text{Embedding3} \)
- \( \text{Proximity4} \rightarrow \text{Embedding4} \)

\[ \ldots \]

- Time consuming!

- \( \text{->} \) How to preserve arbitrary-order proximity while guaranteeing accuracy and efficiency?
Problem Formulation

- **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
    \[ U^* = U \sqrt{\Sigma}, \quad V^* = V \sqrt{\Sigma} \]
Eigen-decomposition Reweighting

- Eigen-decomposition reweighting

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

- **Insights:** high-order proximity is simply re-weighting dimensions!

$$U^* = U\sqrt{\Sigma}, \quad V^* = V\sqrt{\Sigma}$$

Shifting across different orders/weights:

- Preserving arbitrary-order proximity
- Low marginal cost
- Accurate and efficient

Experimental Results

- Link Prediction

Network Structures

Nodes & Links → Pair-wise Proximity → Community Structures → Hyper Edges → Global Structure
Motivation

- Vertexes in different parts of the network may have similar roles (global position)

- Example:
  - Managers in the social network of a company
  - Outliers in a network in the task of anomaly detection

How to reflect the role or importance of a vertex in embedding space?
Existing embedding methods

- They can only preserve local proximity (Structural equivalence), cannot reflect the global position.
- Embeddings of node 5, 6 in the left network will be similar but embeddings of node 1, 2 in the right network will not be similar.
Regular Equivalence

Two nodes are regularly equivalent if their network neighbors are themselves similar (i.e. regularly equivalent).

• Structural equivalence $s$
  • $N(u) = N(v)$
  • Direct way
  • Common neighbors

• Regular equivalence $r$
  • $\{r(i) | i \in N(u)\} = \{r(j) | j \in N(v)\}$
  • Recursive way
  • Similar global position

Regular equivalence is largely ignored in network embedding
Naïve Solutions

- Basis: two regularly equivalent nodes should have similar embeddings
  1. Explicitly calculate the regular equivalence of all vertex pairs
     - infeasible for large-scale networks due to the high complexity of calculating regular equivalence
  2. Replace regular equivalence into simpler graph theoretic metrics
     - centrality measures
     - one centrality can only capture a specific aspect of network role
     - some centrality measures also bear high computational complexity

Deep Recursive Network Embedding

- The definition of regular equivalence is recursive
  - Aggregating neighbors’ information in a recursive way

\[
L_1 = \sum_{v \in V} ||X_v - \text{Agg}({X_u | u \in N(v)})||_F^2,
\]

- How to design the aggregating function
  - Variable length of neighbors
  - Highly nonlinear
  - \rightarrow Layer-normalized LSTM

Deep Recursive Network Embedding

(a) Sampling neighborhoods
(b) Sorting neighborhoods by their degree
(c) Aggregate neighbors
(d) A Weakly guided regularizer

Theoretical Analysis

Theorem 3.5. If the centrality $C(v)$ of node $v$ satisfies that $C(v) = \sum_{u \in N(v)} F(u)C(u)$ and $F(v) = f(\{F(u), u \in N(v)\})$ where $f$ is any computable function, then $C(v)$ is one of the optimal solutions of our model.

<table>
<thead>
<tr>
<th>Centrality</th>
<th>Definition $C(v)$</th>
<th>$F(v)$</th>
<th>$f({x_i})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree</td>
<td>$d_v = \sum_{u \in N(v)} I(d_u)$</td>
<td>$1/d_v$</td>
<td>$1/(\sum I(x_i))$</td>
</tr>
<tr>
<td>Eigenvector</td>
<td>$1/\lambda \times \sum_{u \in N(v)} C(u)$</td>
<td>$1/\lambda$</td>
<td>mean</td>
</tr>
<tr>
<td>PageRank</td>
<td>$\sum_{u \in N(v)} 1/d_u \times C(u)$</td>
<td>$1/d_v$</td>
<td>$1/(\sum I(x_i))$</td>
</tr>
</tbody>
</table>

Experiment --- predict centrality

The MSE value of predicting centralities on Jazz dataset ($10^{-2}$)

<table>
<thead>
<tr>
<th>centrality</th>
<th>closeness</th>
<th>betweenness</th>
<th>eigenvector</th>
<th>k-core</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepWalk</td>
<td>0.6016</td>
<td>3.7188</td>
<td>2.1543</td>
<td>13.2755</td>
</tr>
<tr>
<td>LINE</td>
<td>0.5153</td>
<td>4.3919</td>
<td>1.5072</td>
<td>15.8179</td>
</tr>
<tr>
<td>node2vec</td>
<td>1.0489</td>
<td>3.4065</td>
<td>3.9436</td>
<td>39.2156</td>
</tr>
<tr>
<td>struc2vec</td>
<td>0.2365</td>
<td>0.25371</td>
<td>1.0544</td>
<td>9.0858</td>
</tr>
<tr>
<td>DRNE</td>
<td>0.1909</td>
<td>0.1261</td>
<td>0.5267</td>
<td>5.5683</td>
</tr>
</tbody>
</table>

The MSE value of predicting centralities on BlogCatalog dataset ($10^{-2}$)

<table>
<thead>
<tr>
<th>centrality</th>
<th>closeness</th>
<th>betweenness</th>
<th>eigenvector</th>
<th>k-core</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepWalk</td>
<td>0.2982</td>
<td>1.7836</td>
<td>1.1194</td>
<td>19.7016</td>
</tr>
<tr>
<td>LINE</td>
<td>0.3979</td>
<td>1.8425</td>
<td>1.5167</td>
<td>34.9079</td>
</tr>
<tr>
<td>node2vec</td>
<td>0.3573</td>
<td>1.6958</td>
<td>1.1432</td>
<td>24.1704</td>
</tr>
<tr>
<td>struc2vec</td>
<td>0.2947</td>
<td>1.6018</td>
<td>1.0445</td>
<td>25.3047</td>
</tr>
<tr>
<td>DRNE</td>
<td>0.1101</td>
<td>0.6676</td>
<td>0.3108</td>
<td>7.7210</td>
</tr>
</tbody>
</table>

Experiment - Structural Role Classification

Section Summary

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

Network Characteristics

Application Characteristics
Outline

- Structure-preserved network embedding
- Property-preserved network embedding
- Dynamic network embedding
- Robustness, Explainability and Applicability
- Network embedding for biomedical applications
Why preserve network properties?

Heterogeneity
Transitivity

The Transitivity Phenomenon

Network

A

B

C

Embedding Space

Triangle Inequality: $D(A, B) + D(B, C) > D(A, C)$

A close to B, B close to C, $\rightarrow$ A relatively close to C

However, real network data is complex…
Non-transitivity

The Co-existence of *Transitivity* and *Non-transitivity*

Image network

A
dog
lawn

B
cat
lawn

C
cat
floor

Social network

Colleague
Classmate

Word network

Apple
Cellphone
Banana

How to incorporate non-transitivity in embedding space?
Asymmetric Transitivity

Directed Network

A → B, B → C => A → C, but not C → A

Distance metric in embedding space is symmetric.
How to incorporate Asymmetric Transitivity?
Non-transitivity

The source of non-transitivity:
*Each node has multiple similarity components*

Non-transitive Embedding: represent non-transitive data with multiple latent similarity components

Asymmetric Transitivity

All existing methods fail..

“Asymmetric” fail

“Transitivity” fail

Uncertainties in Networks

- The formation and evolution of real-world networks are full of uncertainties
  - E.g., for the nodes with low degree, they contain less information and thus their representations bear more uncertainties than others.

  - E.g., for the nodes across multiple communities, the possible contradiction between their neighboring nodes may also be large and thus cause the uncertainty.
DVNE for Structure and Uncertainty

Figure 1: The framework of DVNE.

Section Summary

- Compared with network structures, network properties have large space to explore in network embedding.
- Transitivity is important for network inference.
- Uncertainty provides evidence in making network inference.
- Many other property issues:
  - The right embedding space: Euclidean space?
  - Power-law distribution
  - ...
Outline

• Structure-preserved network embedding
• Property-preserved network embedding
• Dynamic network embedding
• Robustness, Explainability and Applicability
• Network embedding for biomedical applications
Dynamic Networks

- Networks are dynamic in nature
  - New (old) nodes are added (deleted)
    - New users, products, etc.
  - The edges between nodes evolve over time
    - Users add or delete friends in social networks, or neurons establish new connections in brain networks.

- How to efficiently incorporate the dynamic changes when networks evolve?
Key problems in dynamic network embedding

- I: Out-of-sample nodes
- II: Incremental edges
- III: Aggregated error
- IV: Scalable optimization
Challenge: High-order Proximity

- High-order proximity
  - Critical structural property of networks
  - Measure indirect relationship between nodes
  - Capture the structure of networks with different scales and sparsity

Network Embedding V.S. Traditional Graph Embedding
Challenge: High-order Proximity

I: Out-of-sample nodes
II: Incremental edges
III: Aggregated error
IV: Scalable optimization

Preserve High-order Proximities

Local Change leads to Global Updating
Key problems in dynamic network embedding

1. Out-of-sample nodes
2. Incremental edges
3. Aggregated error
4. Scalable optimization
Problem

- To infer embeddings for out-of-sample nodes.

- $G = (V, E)$ evolves into $G' = (V', E')$, where $V' = V \cup V^*$.  
- $n$ old nodes: $V = \{v_1, \ldots, v_n\}$, $m$ new nodes: $V^* = \{v_{n+1}, \ldots, v_{n+m}\}$  
- Network embedding: $f: V \to \mathbb{R}^d$  
- We know $f(v)$ for old nodes, want to infer $f(v)$ for new nodes.
Challenges

● Preserve network structures
  ○ e.g. high-order proximity
  ○ need to incorporate prior knowledge on networks

● Share similar characteristics with in-sample embeddings
  ○ e.g. magnitude, mean, variance
  ○ requires a model with great expressive power to fit the data well

● Low computational cost

Specific vs. General

- **Specific**
  - A new NE algorithm capable of handling OOS nodes.

- **General**
  - A solution that helps an arbitrary NE algorithm handle OOS nodes.

- **We propose a general solution.**
  - But it can be easily integrated into an existing NE algorithm (e.g. DeepWalk) to derive a specific algorithm (see the paper).

DepthLGP

- Nonparametric probabilistic modeling + Deep Learning

Design a kernel for the $k$th ($k=1,\ldots,s$) dimension of $h(\cdot)$.

The matrix inversion can be bypassed without approximation.

$a^{(k)}_v$ indicates how much attention we pay to a node. It is learned for an in-sample node, but fixed to one for an OOS node, as we are always interested in OOS nodes.
## Task I: Classification

<table>
<thead>
<tr>
<th>Metric</th>
<th>Embedding</th>
<th>Network</th>
<th>Baselines</th>
<th></th>
<th>This Work</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>LocalAvg</td>
<td>MRG</td>
<td>hLGP</td>
<td>DepthLGP</td>
</tr>
<tr>
<td>Macro-F1(%)</td>
<td>LINE</td>
<td>DBLP</td>
<td>37.89</td>
<td>42.15</td>
<td>47.33</td>
<td>48.25</td>
</tr>
<tr>
<td></td>
<td>PPI</td>
<td></td>
<td>10.52</td>
<td>10.02</td>
<td>13.42</td>
<td>13.72</td>
</tr>
<tr>
<td></td>
<td>BlogCatalog</td>
<td></td>
<td>13.25</td>
<td>11.30</td>
<td>17.41</td>
<td>18.03</td>
</tr>
<tr>
<td>GraRep</td>
<td>DBLP</td>
<td></td>
<td>50.61</td>
<td>55.79</td>
<td>57.43</td>
<td>58.67</td>
</tr>
<tr>
<td></td>
<td>PPI</td>
<td></td>
<td>13.65</td>
<td>13.75</td>
<td>14.80</td>
<td>14.84</td>
</tr>
<tr>
<td></td>
<td>BlogCatalog</td>
<td></td>
<td>14.76</td>
<td>14.80</td>
<td>15.94</td>
<td>18.45</td>
</tr>
<tr>
<td>node2vec</td>
<td>DBLP</td>
<td></td>
<td>53.83</td>
<td>59.34</td>
<td>60.89</td>
<td>62.63</td>
</tr>
<tr>
<td></td>
<td>PPI</td>
<td></td>
<td>15.05</td>
<td>13.43</td>
<td>15.85</td>
<td>16.54</td>
</tr>
<tr>
<td></td>
<td>BlogCatalog</td>
<td></td>
<td>15.10</td>
<td>14.04</td>
<td>19.77</td>
<td>20.32</td>
</tr>
<tr>
<td>Micro-F1(%)</td>
<td>LINE</td>
<td>DBLP</td>
<td>49.58</td>
<td>50.49</td>
<td>54.01</td>
<td>54.94</td>
</tr>
<tr>
<td></td>
<td>PPI</td>
<td></td>
<td>18.10</td>
<td>15.71</td>
<td>20.71</td>
<td>21.42</td>
</tr>
<tr>
<td></td>
<td>BlogCatalog</td>
<td></td>
<td>27.40</td>
<td>23.21</td>
<td>31.36</td>
<td>31.90</td>
</tr>
<tr>
<td>GraRep</td>
<td>DBLP</td>
<td></td>
<td>60.17</td>
<td>60.62</td>
<td>61.44</td>
<td>62.29</td>
</tr>
<tr>
<td></td>
<td>PPI</td>
<td></td>
<td>20.23</td>
<td>20.35</td>
<td>20.79</td>
<td>21.44</td>
</tr>
<tr>
<td></td>
<td>BlogCatalog</td>
<td></td>
<td>36.44</td>
<td>30.79</td>
<td>37.57</td>
<td>38.14</td>
</tr>
<tr>
<td>node2vec</td>
<td>DBLP</td>
<td></td>
<td>60.54</td>
<td>62.29</td>
<td>62.83</td>
<td>64.56</td>
</tr>
<tr>
<td></td>
<td>PPI</td>
<td></td>
<td>19.70</td>
<td>18.25</td>
<td>22.63</td>
<td>23.11</td>
</tr>
<tr>
<td></td>
<td>BlogCatalog</td>
<td></td>
<td>34.83</td>
<td>25.82</td>
<td>37.96</td>
<td>39.64</td>
</tr>
</tbody>
</table>

Key problems in dynamic network embedding

- I : Out-of-sample nodes
- II : Incremental edges
- III: Aggregated error
- IV: Scalable optimization
The Static Model

- We aim to preserve high-order proximity in the embedding matrix with the following objective function:

  \[
  \min \| S - UU'^T \|_F^2
  \]

  - where \( S \) denotes the high-order proximity matrix of the network
  - \( U \) and \( U' \) is the results of matrix decomposition of \( S \).

- For undirected networks, \( U \) and \( U' \) are highly correlated.
  - Without loss of generality, we choose \( U \) as the embedding matrix.

We choose Katz Index as $S$ because it is one of the most widely used measures of high-order proximity.

$$S^{Katz} = M_a^{-1}M_b$$

$$M_a = (I - \beta A)$$

$$M_b = \beta A$$

where $\beta$ is a decay parameter, $I$ is the identity matrix and $A$ is the adjacency matrix.

According to HOPE, the original objective function can be solved by the generalized SVD (GSVD) method.
Generalized Eigen Perturbation

- We propose generalized eigen perturbation to fulfill the task.
  - The goal of generalized eigen perturbation is to update $X(t)$ to $X(t+1)$

- Specifically, given the change of adjacency matrix $\Delta A$ between two consecutive time steps, the change of $M_a$ and $M_b$ can be represented as:

$$\Delta M_a = -\beta \Delta A, \text{ and } \Delta M_b = \beta \Delta A$$

Key problems in dynamic network embedding

- I: Out-of-sample nodes
- II: Incremental edges
- III: Aggregated error
- IV: Scalable optimization
Problem: Error Accumulation

- Eigen perturbation is at the cost of inducing approximation

\[ A_0 \xrightarrow{SVD} \begin{bmatrix} U_0 & \Sigma_0 & V_0 \end{bmatrix} \]

\[ A_1 \approx \begin{bmatrix} U_1' & \Sigma_1' & V_1' \end{bmatrix} \]

\[ \ldots \approx \begin{bmatrix} U_t' & \Sigma_t' & V_t' \end{bmatrix} \]

- Problem: error accumulation is inevitable
Solution: SVD Restarts

• Solution: restart SVD occasionally

  \[
  A_0 \xrightarrow{SVD} U_0 \Sigma_0 V_0 \\
  A_1 \xrightarrow{SVD} U_1' \Sigma_1' V_1' \\
  \vdots
  \]

  \[
  A_t \xrightarrow{SVD} U_t \Sigma_t V_t \\
  \]

  \[
  A_{t+1} \xrightarrow{SVD} U_{t+1}' \Sigma_{t+1}' V_{t+1}' \\
  \]

  \[
  \Rightarrow \text{Update} \\
  \Rightarrow \text{Update}
  \]

• What are the appropriate time points?
  • Too early restarts: waste of computation resources
  • Too late restarts: serious error accumulation
Naïve Solution

- Naïve solution: fixed time interval or fixed number of changes
- Difficulty: error accumulation is not uniform

Existing Method

- Existing method: monitor loss (Chen and Candan, KDD 2014)
- Loss in SVD:
  \[ J = \| S - U\Sigma V^T \|_F^2 \]
  \( S \): target matrix, \([U, \Sigma, V]\): results of SVD
- Problem: loss includes approximation error and intrinsic loss in SVD

### Framework: Monitor Margin

- **Observation:** the margin between the current loss and intrinsic loss in SVD is the actual accumulated error
  - **Current loss:** $J = \| S - U\Sigma V^T \|_F^2$
  - **Intrinsic loss:** $\mathcal{L}(S, k) = \min_{U^*,\Sigma^*,V^*} \| S - U^*\Sigma^*V^*^T \|_F^2$, $k$: dimensionality

---

Solution: Lazy Restarts

- Lazy restarts: restart only when the margin exceeds the threshold
- Problem: intrinsic loss is hard to compute
  - Direct calculation has the same time complexity as SVD
- Relaxation: an upper bound on margin
  - A lower bound on intrinsic loss $\mathcal{L}(S,k)$

\[
\mathcal{L}(S_t, k) \geq B(t) \Rightarrow \frac{\mathcal{J}(t) - \mathcal{L}(S_t, k)}{\mathcal{L}(S_t, k)} \leq \frac{\mathcal{J}(t) - B(t)}{B(t)}.
\]

$\mathcal{J}(t)$: current loss; $\mathcal{L}(S_t,k)$: intrinsic loss; $B(t)$: bound of intrinsic loss

A Lower Bound of SVD Intrinsic Loss

- Idea: use matrix perturbation

**Theorem 1** (A Lower Bound of SVD Intrinsic Loss). If $S$ and $\Delta S$ are symmetric matrices, then:

$$
\mathcal{L}(S + \Delta S, k) \geq \mathcal{L}(S, k) + \Delta tr^2(S + \Delta S, S) - \sum_{l=1}^{k} \lambda_l, \quad (9)
$$

where $\lambda_1 \geq \lambda_2 \ldots \geq \lambda_k$ are the top-$k$ eigenvalues of $\nabla_{S^2} = S \cdot \Delta S + \Delta S \cdot S + \Delta S \cdot \Delta S$, and

$$
\Delta tr^2(S + \Delta S, S) = tr((S + \Delta S) \cdot (S + \Delta S)) - tr(S \cdot S).
$$

- Intuition: treat changes as a perturbation to the original network

Time Complexity Analysis

**Theorem 2.** The time complexity of calculating $B(t)$ in Eqn (13) is $O(M_S + M_L k + N_L k^2)$, where $M_S$ is the number of the non-zero elements in $\Delta S$, and $N_L, M_L$ are the number of the non-zero rows and elements in $\nabla S^2$ respectively.

- If every node has a equal probability of adding new edges, we have: $M_L \approx 2d_{avg}M_S$, where $d_{avg}$ is the average degree of the network.
- For Barabasi Albert model (Barabasi and Albert 1999), a typical example of preferential attachment networks, we have: $M_L \approx \frac{12}{\pi^2} [\log(d_{max}) + \gamma] M_S$, where $d_{max}$ is the maximum degree of the network and $\gamma \approx 0.58$ is a constant.

**Conclusion:** the complexity is only linear to the local dynamic changes

Experimental Results: Approximation Error

- Fixing number of restarts

<table>
<thead>
<tr>
<th>Dataset</th>
<th>avg(r)</th>
<th>max(r)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIMERS</td>
<td>LWI2</td>
</tr>
<tr>
<td>FACEBOOK</td>
<td>0.005</td>
<td>0.020</td>
</tr>
<tr>
<td>MATH</td>
<td>0.037</td>
<td>0.057</td>
</tr>
<tr>
<td>WIKI</td>
<td>0.053</td>
<td>0.086</td>
</tr>
<tr>
<td>DBLP</td>
<td>0.042</td>
<td>0.110</td>
</tr>
<tr>
<td>INTERNET</td>
<td>0.152</td>
<td>0.218</td>
</tr>
</tbody>
</table>

- Fixing maximum error

27%~42% Improvement

Experimental Results: Analysis

- Syntactic networks: simulate drastic changes in the network structure

- Robust to sudden changes

- Linear scalability
Key problems in dynamic network embedding

- I: Out-of-sample nodes
- II: Incremental edges
- III: Aggregated error
- IV: Scalable optimization
Highly-dynamic & Recency-sensitive Data

- News recommendation applications: a bipartite graph
- WeChat news recommendation network is highly dynamic
  - 81 articles and 1400 reading records per second
- The network is also recency-sensitive
  - >73% articles died less than 6 hours while no one read again
  - Obvious exponential decay for article duration length.

Limited resources

- We cannot guarantee convergence in-between every two timestamps.
- Just do it.

- How to do better?
- Non-uniform resource allocation.
- New edges and nodes worth more resources.

Diffused SGD: Step-wise Weight Diffusion Mechanism

- The Change of a node embedding vector depends on its distance to the changed edge.
- Diffuse across training steps
- For step $r$, if edge $(i, j)$ is chosen by stochastic method

For edge $(i, j)$, we have

$$p_{i,j}(r) \leftarrow \tau_e (i, p_{i,j}(r-1));$$

for $(i, k) \in E \land k \neq j$, we use

$$p_{i,k}(r) \leftarrow p_{i,k}(r-1) + \tau_n (i, p_{i,j}(r-1));$$

and for other edges $(l, k) \in E \land l \neq i$,

$$p_{l,k}(r) \leftarrow p_{l,k}(r-1);$$

Section Summary

● I: Out-of-sample nodes
  ○ DepthLGP = Non-parametric GP + DNN

● II: Incremental edges
  ○ DHPE: Generalized Eigen Perturbation

● III: Aggregated error
  ○ TIMERS: A theoretically guaranteed SVD restart strategy

● IV: Scalable optimization
  ○ D-SGD: A iteration-wise weighted SGD for highly dynamic data
Outline

- Structure-preserved network embedding
- Property-preserved network embedding
- Dynamic network embedding
- Robustness, Explainability and Applicability
- Network embedding for biomedical applications
Technical challenges in real applications

- Robustness
- Interpretability
- Applicability

Hot directions in computer vision:

- Adversarial
- Explainable
- Scalable
Robustness in network embedding

- Adversarial attacks
- Small perturbations in graph structures and node attributes
- Great challenges for applying GCNs to node classification
Adversarial Attacks on GCNs

Categories

- Targeted VS Non-targeted
  - Targeted: the attacker focus on misclassifying some target nodes
  - Non-targeted: the attacker aims to reduce the overall model performance

- Direct vs Influence
  - Direct: the attacker can directly manipulate the edges or features of the target nodes
  - Influence: the attacker can only manipulate other nodes except the targets

How to enhance the robustness of GCNs against adversarial attacks?
Robust Graph Convolutional Networks

- Adversarial attacks in node classification
  - Connect nodes from different communities to confuse the classifier

- Distribution V.S. plain vectors
  - Plain vectors cannot adapt to such changes
  - Variances can help to absorb the effects of adversarial changes
  - Gaussian distributions -> Hidden representations of nodes

The Framework of RGCN

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

Experimental Results

- 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.

Interpretability of network embedding

- 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**.
Disentangled Representation Learning

- That is, we aim to learn disentangled node representation,
  - A representation that contains independent components, that describes different aspects (caused by different latent factors) of the observation.
- The topic is well studied in the field of computer vision.
  - But largely unexplored in the literature of GNNs.

Example: Three dimensions that are related skin color, age/gender, and saturation, respectively.
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.

A neighbor is **patched to channel** $k$ (for further in-channel graph convolution), if the edge between the neighbor and the center node is **caused by factor** $k$.

But the actual causes are unknown. **Neighborhood routing** is therefore proposed to infer the latent causes, based on two hypothesis.

- The first is analogous to the second-order proximity.

  **Hypothesis 1.** Factor $k$ is likely to be the reason why node $u$ connects with a certain subset of its neighbors, if the subset is large and the neighbors in the subset are similar w.r.t. aspect $k$, i.e., they form a cluster in the $k^{th}$ subspace.

- It inspires us to search for the biggest cluster in each of the $K$ subspaces.

---

Neighborhood Routing: Hypothesis II

- The second hypothesis is analogous to the first-order proximity.

  **Hypothesis 2.** Factor $k$ is likely to be the reason why node $u$ and neighbor $v$ are connected, if the two are similar in terms of aspect $k$.

- Hypothesis 2 is not robust if either $x_u$ or $x_v$ misses features about aspect $k$, and therefore must be combined with Hypothesis 1. But it can provide a fast guess.

Results: Multi-label Classification

(a) Macro-F1(%), BlogCatalog.

(b) Micro-F1(%), BlogCatalog.

(c) Macro-F1(%), PPI.

(d) Micro-F1(%), PPI.

(e) Macro-F1(%), POS.

(f) Micro-F1(%), POS.

Figure 2. Macro-F1 and Micro-F1 scores on the multi-label classification tasks. Our approach consistently outperforms the best performing baselines by a large margin, reaching 10% to 20% relative improvement in most cases.

## Results: On Synthetic Graphs

<table>
<thead>
<tr>
<th>Method</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>14</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>78.78</td>
<td>65.73</td>
<td>46.55</td>
<td>37.37</td>
<td>24.49</td>
<td>18.14</td>
<td>16.43</td>
</tr>
<tr>
<td>GAT</td>
<td>83.77</td>
<td>60.89</td>
<td>45.88</td>
<td>36.72</td>
<td>24.77</td>
<td>20.89</td>
<td>19.53</td>
</tr>
<tr>
<td>DisenGCN (this work)</td>
<td><strong>93.84</strong></td>
<td><strong>74.68</strong></td>
<td><strong>54.57</strong></td>
<td><strong>43.96</strong></td>
<td><strong>28.17</strong></td>
<td><strong>23.57</strong></td>
<td><strong>21.99</strong></td>
</tr>
<tr>
<td>Relative improvement</td>
<td>+12.02%</td>
<td>+13.62%</td>
<td>+17.23%</td>
<td>+17.63%</td>
<td>+13.73%</td>
<td>+12.83%</td>
<td>+12.6%</td>
</tr>
</tbody>
</table>

- Improvement is larger when `#factors` is relatively large (around 8).
- However, all methods are bad when `#factors` is extremely large.

Results: Correlations between the Neurons

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

Applicability of network embedding

Various network properties

- High-order
- Global position

Various applications

- Link Prediction
- Community Detection
- Node Classification
- Network Distance
- Node Importance
- …

Leading to a large number of hyperparameters
Must be carefully tuned

AutoML
AutoML

• Ease the adoption of machine learning and reduce the reliance on human experts
  • e.g., hyperparameter optimization
• Largely unexplored on network data

• Large scale issue:
  • Complexity of Network Embedding is usually at least $O(E)$
    • $E$ is the number of edges (can be 10 billion)
  • Total complexity: $O(ET)$, $T$ is the times searching for optimal hyperparameters

How to incorporate AutoML into massive network embedding efficiently?
AutoML for network embedding

• A straightforward way: configuration selection on sampled sub-networks

• Transferability
  • $\theta \neq$ optimal configuration on origin network

• Heterogeneity
  • several highly heterogeneous components $\Rightarrow$ carefully designed sampling

AutoNE

Transfer the knowledge about optimal hyperparameters from the sub-networks to the original massive network

Experiment --- Sampling-Based NE

The performance achieved within various time thresholds.

The number of trials to reach a certain performance threshold

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.

Scalability and parallelization. In the big-data era, real graphs can easily have millions of nodes and edges, such as social networks or e-commerce networks [8]. As a result, how to design scalable models, preferably with a linear time complexity, becomes a key problem. In addition, since nodes and edges in the graph are interconnected and often need to be modeled as a whole, how to conduct parallel computing is another critical issue.
Outline

- Structure-preserved network embedding
- Property-preserved network embedding
- Dynamic network embedding
- Robustness, Explainability and Applicability
- Network embedding for biomedical applications
Network Embedding for Biomedical Applications

Network embedding in biomedicine

Pharmaceutical data analysis
- Drug repositioning
- Adverse drug reaction analysis

Multi-omics data analysis
- Genomics
- Proteomics
- Transcriptomics

Clinical data analysis
- Medical knowledge graph embedding
- EHR embedding

1. Chang Su, Jie Tong, Yongjun Zhu, Peng Cui, Fei Wang, Network embedding in biomedical data science, Briefings in Bioinformatics, bby117, https://doi.org/10.1093/bib/bby117
Pharmaceutical data analysis

Drug repositioning

Exploring new usage for existing drugs.

- save drug development cost
- increase productivity

Aiming at predicting:

- unknown drug-target interactions
- unknown drug-disease interactions
Pharmaceutical data analysis

- Drug-target interaction prediction

Matrix factorization based embedding methods, e.g., LE

Drug-target interaction prediction (Zong et al. 2017)

Pharmaceutical data analysis

- Drug-disease interaction prediction (Dai et al. 2015)

Pharmaceutical data analysis

- Drug-disease interaction prediction (Wang et al. 2017)

Pharmaceutical data analysis

- Adverse drug reaction analysis

An adverse drug reaction (ADR) is defined as any undesirable effect from the medical use of drugs beyond its anticipated therapeutic effects that occurs at a usual dosage.

The ADR study is implemented before a drug is launched on clinical application.

- Adverse drug reaction (ADR) prediction
- Drug-drug interaction (DDI) prediction
Pharmaceutical data analysis

- Adverse drug reaction (ADR) prediction (Stanovsky et al. 2017)

Identify ADR from social media posts:

Output labels of each word, B, I, and O:
(B) Beginning of an ADR span; (I), Inside an ADR span; (O), Out-side of the span of an ADR

Aspirin made me feel dizzy

Pharmaceutical data analysis

- Drug-drug interaction (DDI) prediction (Abdelaziz et al. 2017)

Pharmaceutical data analysis

- Drug-drug interaction (DDI) prediction (Wang et al. 2017)


Heterogeneous medical knowledge graph

Basic triple: \((h, r, t)\)
- \(h, t\): entities (drug, protein, pathway, phenotype)
- \(r\): relation

Existing drug-drug interactions

Rich DDI triple: \((u, l, v)\)
- \(u, v\): drug entities
- \(l\): text label of DDI

Basic triple encoder:
\[ z_{bte}(h, r, t) = b_1 - \|hM_r + r - tM_r\|_{L1/L2} \]

Rich DDI triple encoder:
\[ z_{dte}(u, l, v) = b_2 - \|uM_l + l - vM_l\|_{L1/L2} \]

Joint learning:
\[ O(X) = \mathcal{L}_{bte} + \mathcal{L}_{dte} + \mathcal{L}_{rel} + \gamma C(X) \]

New DDIs

Text label encoder

Regularization term
Multi-omics data analysis

- **Genomics data analysis**
  - Gene function prediction (Wang et al. 2015)

Multi-omics data analysis

- Identification of Pathways Associated with Chemosensitivity (Wang et al. 2017)

Multi-omics data analysis

- Cell and gene representation (Li et al. 2017)

Multi-omics data analysis

- **Proteomics data analysis**
  - Protein-protein interaction (PPI) prediction (Cannistraci et al. 2013)

  Experimental identification of PPIs is time-consuming. Computational method was proposed to predict candidate PPIs based on existing interactions.

Multi-omics data analysis

- Protein function prediction (Wang et al. 2017)

1. Construct protein-protein network;
2. Learn low-dimensional representations for proteins;
3. Calculate an intra-species affinity score and an inter-species affinity score by transferring Gene annotations;
4. Rank the score and pick the function(s) with the highest score(s) for queried protein(s)

Multi-omics data analysis

- Transcriptomics data analysis
  - MicroRNA-disease association prediction (Li et al. 2017)
    - microRNAs (miRNAs) are connected with several complex human diseases;
    - Identifying human disease-related miRNAs will be useful in uncovering novel prognostic markers for cancer.

![Diagram showing network embedding, similarity learning, and prediction of unknown associations.]

Medical data analysis

- Medical knowledge graph embedding
  - Safe medication recommendation (Wang et al. 2017) – predicting patient-medicine association

\[
\mathcal{L}(X) = \mathcal{L}_{G_m} + \mathcal{L}_{G_d} + \mathcal{L}_{G_{pm}} + \mathcal{L}_{G_{pd}} + \gamma C(X)
\]

Medical data analysis

- Representation learning on medical forum data (Zhao et al. 2017)

Learning low-dimensional representations of diseases and symptoms.

---

Medical data analysis

- **EHR embedding**
  - **Healthcare representation learning (Choi et al. 2017)**
    - EHR representation with hierarchical information inherent to medical ontologies;
    - The model represents a medical concept as a combination of its ancestors in the ontology via an attention mechanism.

Network Embedding for Biomedical Applications

- **Challenges**
  - Data quality. Networks constructed from the biomedical data are usually noisy and incomplete. For example, the PPI data produced by high-throughput techniques, suffer from high false negative rates up to 70% and high false positive rates up to 64%.
  - Local and global. Network embedding and its downstream tasks rely on the type of structural property to preserve. Designing embedding method by properly considering local and global structure properties require the development of novel solutions.
  - Network evolution. Networks are always not static, especially in the biomedical domain. Yet most existing embedding models focus on static network.
  - Domain complexity. Network structure is highly associated with domain knowledge.
Network Embedding for Biomedical Applications

- Opportunities
  - Local and global trade-off embedding.
  - Dynamic network embedding.
  - Text associated embedding. Medical knowledge bases always contain rich text information such as descriptions of entities and relations, which would have high potential to address network incompleteness and improve understanding of topological properties.
  - Domain-knowledge-associated embedding. Incorporating external domain knowledge into network embedding.
Summary and Conclusion

- Structure-preserved network embedding
  - Guarantee information equivalence
- Property-preserved network embedding
  - Enabling network inference in embedding space
- Dynamic network embedding
  - Incorporating dynamic changes
- Robustness, Explainability and Applicability
  - Promote network embedding in real applications
- Network embedding for biomedical applications
  - Prove the effectiveness of NE in real applications