Part III: Network Embedding with Deep Learning

Xin Zhao
batmanfly@qq.com
Renmin University of China
Outline

• Preliminaries
  – word2vec

• Network Embedding Models
  – DeepWalk
  – Node2vec
  – GENE
  – LINE
  – SDNE

• Applications of Network Embedding
  – Basic applications
  – Visualization
  – Text classification
  – Recommendation

• Conclusion
Preliminaries

• Softmax functions
• Distributional semantics
• Word2vec
  – CBOW
  – Skip-gram
Preliminaries

• Representation learning
  – Using machine learning techniques to derive data representation

• Distributed representation
  – Different from one-hot representation, it uses dense vectors to represent data points

• Embedding
  – Mapping information entities into a low-dimensional space
Softmax function

• It transforms a $K$-dimensional real vector into a *probability distribution*

  – A common transformation function to derive objective functions for classification or discrete variable modeling

$$
\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K
$$
he curtains open and the stars shining in on the barely
ars and the cold, close stars " . And neither of the w
rough the night with the stars shining so brightly, it
made in the light of the stars. It all boils down, wr
surely under the bright stars, thrilled by ice-white
sun, the seasons of the stars? Home, alone, Jay pla
m is dazzling snow, the stars have risen full and cold
un and the temple of the stars, driving out of the hug
in the dark and now the stars rise, full and amber a
bird on the shape of the stars over the trees in front
But I could n’t see the stars or the moon, only the
they love the sun, the stars and the stars. None of
r the light of the shiny stars. The splash of flowing w
man ’s first look at the stars; various exhibits, aer
rief information on both stars and constellations, inc
Distributional semantics

- Collect the contextual words for “stars”

<table>
<thead>
<tr>
<th>Construct vector representations</th>
<th>shining</th>
<th>bright</th>
<th>trees</th>
<th>dark</th>
<th>look</th>
</tr>
</thead>
<tbody>
<tr>
<td>stars</td>
<td>38</td>
<td>45</td>
<td>2</td>
<td>27</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Similarity in meaning as vector similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>• cucumber</td>
</tr>
<tr>
<td>• stars</td>
</tr>
<tr>
<td>• sun</td>
</tr>
</tbody>
</table>
Word2Vec

• Input: a sequence of words from a vocabulary \( V \)

• Output: a fixed-length vector for each term in the vocabulary
  \(- v_w\)

It implements the idea of distributional semantics using a shallow neural network model.
Architecture 1: CBOW

- **CBOW** predicts the current word using surrounding contexts
  - \( Pr(w_t | context(w_t)) \)

- Window size 2c
- \( context(w_t) = [w_{t-c}, ..., w_{t+c}] \)
Architecture 1: CBOW

- **CBOW** predicts the current word using surrounding contexts
  - \( Pr(w_t | context(w_t)) \)

- Using a \( K \)-dimensional vector to represent words
  - \( w_t \rightarrow \nu_{w_t} \)
  - \( \tilde{\nu}_{w_t} = \frac{\sum_{i=t-c}^{t+c} \nu_{w_i}}{2c} \quad (i \neq t) \)
Architecture 1: CBOW

- **CBOW** predicts the current word using surrounding contexts
  
  \( Pr(w_t | \text{context}(w_t)) \)

- Basic Idea
  
  - Given the context of the current word \( \tilde{v}_{w_t} \)
  
  - \( \text{Sim}(\tilde{v}_{w_t}, v_{w_t}) > \text{Sim}(\tilde{v}_{w_t}, v_{w_j}) \)
Architecture 1: CBOW

• How to formulate the idea
  – Using a softmax function
  – Considered as a classification problem
    • Each word is a classification label

\[ P(w | w_{\text{context}}) = \frac{\exp(sim(\tilde{v}_w, v_w))}{\sum_{w'} \exp(sim(\tilde{v}_w, v_{w'}))} \]
Architecture 2

- **Skip-gram** predicts surrounding words using the current word
  
  \[ \text{Pr}(\text{context}(w_t) \mid w_t) \]
  
  - Window size 2c
  - \( \text{context}(w_t) = [w_{t-c}, \ldots, w_{t+c}] \)
Architecture 2

- **Skip-gram** predicts surrounding words using the current word
  
  \[
  Pr(\text{context}(w_t) \mid w_t)
  \]

  - Window size 2c
  - \(\text{context}(w_t) = [w_{t-c}, ..., w_{t+c}]\)

  \[
  P(w' \mid w) = \frac{\exp(sim(v_w, v_{w'}))}{\sum_{w''} \exp(sim(v_w, v_{w''}))}
  \]
Network Embedding Models

- DeepWalk
- Node2vec
- GENE
- LINE
- SDNE
Network Embedding Models

- **DeepWalk** (Perozzi et al., KDD 2014)
- Node2vec
- GENE
- LINE
- SDNE
What is network embedding?

• We map each node in a network into a low-dimensional space
  – Distributed representation for nodes
  – Similarity between nodes indicate the link strength
  – Encode network information and generate node representation
Example

- Zachary’s Karate Network:
DeepWalk

- DeepWalk learns a latent representation of adjacency matrices using deep learning techniques developed for language modeling
Language modeling

• Learning a representation of a word from documents (word co-occurrence):
  – word2vec: \( \Phi: v \in V \mapsto \mathbb{R}^{|V| \times d} \)

• The learned representations capture inherent structure

• Example:

\[
\|\Phi(\text{rose}) - \Phi(\text{daisy})\| < \|\Phi(\text{rose}) - \Phi(\text{tiger})\|
\]
From language modeling to graphs

• Idea:
  – Nodes --> Words
  – Node sequences --> Sentences

• Generating node sequences:
  – Using random walks
    • short random walks = sentences

• Connection:
  – **Words frequency** in a natural language corpus follows a power law.
  – **Vertex frequency** in random walks on scale free graphs also follows a power law.
Framework

1. Input: Graph

2. Random Walks

3. Representation Mapping

4. Hierarchical Softmax

5. Output: Representation
Representation Mapping

\[ W_{v_4} = 4 \]

- Map the vertex under focus \((v_1)\) to its representation.
- Define a window of size \(W\).
- If \(W = 1\) and \(v = v_1\)

Maximize:

\[
\begin{align*}
\Pr(v_3 | \Phi(v_1)) \\
\Pr(v_5 | \Phi(v_1))
\end{align*}
\]
Deep Learning Structure: Skip-gram model

Skip-gram: The input to the model is $w_i$, and the output could be $w_{i-1}, w_{i-2}, w_{i+1}, w_{i+2}$

Maximize: $\Pr(v_3 | \Phi(v_1))$

$\Pr(v_5 | \Phi(v_1))$
Experiments

• Node Classification
  – Some nodes have labels, some don’t

• DataSet
  – BlogCatalog
  – Flickr
  – YouTube
## Results: BlogCatalog

<table>
<thead>
<tr>
<th>% Labeled Nodes</th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
<th>40%</th>
<th>50%</th>
<th>60%</th>
<th>70%</th>
<th>80%</th>
<th>90%</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DEEPWALK</strong></td>
<td>36.00</td>
<td>38.20</td>
<td>39.60</td>
<td>40.30</td>
<td>41.00</td>
<td>41.30</td>
<td>41.50</td>
<td>41.50</td>
<td>42.00</td>
</tr>
<tr>
<td>SpectralClustering</td>
<td>31.06</td>
<td>34.95</td>
<td>37.27</td>
<td>38.93</td>
<td>39.97</td>
<td>40.99</td>
<td>41.66</td>
<td>42.42</td>
<td>42.62</td>
</tr>
<tr>
<td>EdgeCluster</td>
<td>27.94</td>
<td>30.76</td>
<td>31.85</td>
<td>32.99</td>
<td>34.12</td>
<td>35.00</td>
<td>34.63</td>
<td>35.99</td>
<td>36.29</td>
</tr>
<tr>
<td>Modularity</td>
<td>27.35</td>
<td>30.74</td>
<td>31.77</td>
<td>32.97</td>
<td>34.09</td>
<td>36.13</td>
<td>36.08</td>
<td>37.23</td>
<td>38.18</td>
</tr>
<tr>
<td>wvRN</td>
<td>19.51</td>
<td>24.34</td>
<td>25.62</td>
<td>28.82</td>
<td>30.37</td>
<td>31.81</td>
<td>32.19</td>
<td>33.33</td>
<td>34.28</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Micro-F1(%)</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DEEPWALK</strong></td>
<td>21.30</td>
<td>23.80</td>
<td>25.30</td>
<td>26.30</td>
<td>27.30</td>
<td>27.60</td>
<td>27.90</td>
<td>28.20</td>
<td>28.90</td>
</tr>
<tr>
<td>SpectralClustering</td>
<td>19.14</td>
<td>23.57</td>
<td>25.97</td>
<td>27.46</td>
<td>28.31</td>
<td>29.46</td>
<td>30.13</td>
<td>31.38</td>
<td>31.78</td>
</tr>
<tr>
<td>EdgeCluster</td>
<td>16.16</td>
<td>19.16</td>
<td>20.48</td>
<td>22.00</td>
<td>23.00</td>
<td>23.64</td>
<td>23.82</td>
<td>24.61</td>
<td>24.92</td>
</tr>
<tr>
<td>Modularity</td>
<td>17.36</td>
<td>20.00</td>
<td>20.80</td>
<td>21.85</td>
<td>22.65</td>
<td>23.41</td>
<td>23.89</td>
<td>24.20</td>
<td>24.97</td>
</tr>
<tr>
<td>wvRN</td>
<td>6.25</td>
<td>10.13</td>
<td>11.64</td>
<td>14.24</td>
<td>15.86</td>
<td>17.18</td>
<td>17.98</td>
<td>18.86</td>
<td>19.57</td>
</tr>
<tr>
<td>Majority</td>
<td>2.52</td>
<td>2.55</td>
<td>2.52</td>
<td>2.58</td>
<td>2.58</td>
<td>2.63</td>
<td>2.61</td>
<td>2.48</td>
<td>2.62</td>
</tr>
</tbody>
</table>
Network Embedding Models

- DeepWalk
- **Node2vec** *(Grover et al., KDD 2016)*
- GENE
- LINE
- SDNE
Node2Vec

• A generalized version of DeepWalk
  – Objective function
    \[
    \max_f \sum_{u \in V} \log Pr(N_S(u)|f(u)).
    \]
  – Conditional independence
    \[
    Pr(N_S(u)|f(u)) = \prod_{n_i \in N_S(u)} Pr(n_i|f(u)).
    \]
  – Symmetry in feature space
    \[
    Pr(n_i|f(u)) = \frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))}.
    \]
Node2Vec

$N_S(u) \subset V$

– a network neighborhood of node $u$ generated through a neighborhood sampling strategy $S$.

– The key lies in how to find a neighbor on the graph

– How DeepWalk solve this?

$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v, x) \in E \\ 0 & \text{otherwise} \end{cases}$$

where $\pi_{vx}$ is the unnormalized transition probability between nodes $v$ and $x$, and $Z$ is the normalizing constant.
How Node2vec Do this?

• Motivation

Figure 1: BFS and DFS search strategies from node $u$ ($k = 3$).

– BFS: broader $\rightarrow$ **homophily**
– DFS: deeper $\rightarrow$ **structural equivalence**
How Node2vec Do this?

• Can we combine the merits of DFS and BFS
  – BFS: broader $\rightarrow$ **homophily**
  – DFS: deeper $\rightarrow$ **structural equivalence**

\[
P(c_i = x | c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v, x) \in E \\ 0 & \text{otherwise} \end{cases}
\]

\[
\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}
\]

\[
\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}
\]
How Node2vec Do this?

- Explaining the sampling strategy

\[
P(c_i = x \mid c_{i-1} = v) = \begin{cases} 
\frac{\pi_{vx}}{Z} & \text{if } (v, x) \in E \\
0 & \text{otherwise}
\end{cases}
\]

\[
\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}
\]

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

**Return parameter, p.** Parameter p controls the likelihood of immediately revisiting a node in the walk.

**In-out parameter, q.** Parameter q allows the search to differentiate between “inward” and “outward” nodes.
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 $q$)

$\pi = \text{PreprocessModifiedWeights}(G, p, q)$

$G' = (V, E, \pi)$

Initialize walks to Empty

for $iter = 1$ to $r$ do

   for all nodes $u \in V$ do

      $walk = \text{node2vecWalk}(G', u, l)$

      Append $walk$ to walks

   $f = \text{StochasticGradientDescent}(k, d, walks)$

return $f$

node2vecWalk (Graph $G' = (V, E, \pi)$, Start node $u$, Length $l$)

Initialize $walk$ to $[u]$

for $walk_{iter} = 1$ to $l$ do

   $curr = walk[-1]$

   $V_{curr} = \text{GetNeighbors}(curr, G')$

   $s = \text{AliasSample}(V_{curr}, \pi)$

   Append $s$ to $walk$

return $walk$
Comparison between DeepWalk and Node2vec

• They actually have the same objective function and formulations

• The difference lies in how to generate random walks

• BEAUTY: node $\rightarrow$ word, path $\rightarrow$ sentence
Network Embedding Models

• DeepWalk
• Node2vec
• GENE (Chen et al., CIKM 2016)
• LINE
• SDNE
GENE

• Incorporate Group Information to Enhance Network Embedding
  – When group information is available, how to model it?
    • Group $\rightarrow_{\text{control}}$ member
GENE

• Recall doc2vec

• How to use doc2vec to model group and member vectors
GENE

• Incorporate Group Information to Enhance Network Embedding
  – When group information is available, how to model it?
GENE

• Formulate the idea

\[
\mathcal{L} = \sum_{g_i \in C} \left( \alpha \sum_{W \in W_{g_i}} \sum_{v_j \in W} \log p(v_j | v_{j-k}, \ldots, v_{j+k}, g_i) \right) + \\
\beta \sum_{\hat{v}_j \in W_{g_i}} \log p(\hat{v}_j | g_i),
\]

(1)

\[
\log p(v_j | v_{j-k}, \ldots, v_{j+k}, g_i) = \frac{\exp(\bar{u}^T u'_j)}{\sum_{n=1}^M \exp(\bar{u}^T u'_n)},
\]

(2)

\[
\log p(\hat{v}_j | g_i) = \frac{\exp(u_{g_i}^T \hat{u}_j)}{\sum_{n=1}^M \exp(u_{g_i}^T \hat{u}_n)},
\]

(3)
Network Embedding Models

- DeepWalk
- Node2vec
- GENE
- **LINE** (Tang et al., WWW 2015)
- SDNE
LINE

First-order Proximity

- The local pairwise proximity between the vertices
  - Determined by the observed links
- However, many links between the vertices are missing
  - Not sufficient for preserving the entire network structure

Vertex 6 and 7 have a large first-order proximity
Second-order Proximity

• The proximity between the *neighborhood structures* of the vertices

• Mathematically, the second-order proximity between each pair of vertices \((u,v)\) is determined by:

\[
\hat{p}_u = (w_{u1}, w_{u2}, ..., w_{u|V|}) \\
\hat{p}_v = (w_{v1}, w_{v2}, ..., w_{v|V|})
\]

Vertex 5 and 6 have a large second-order proximity

\[
\hat{p}_5 = (1,1,1,0,0,0,0,0,0,0) \\
\hat{p}_6 = (1,1,1,0,0,5,0,0,0,0)
\]
LINE

Preserving the First-order Proximity

• Given an undirected edge \((v_i, v_j)\), the joint probability of \(v_i, v_j\)

\[
p_1(v_i, v_j) = \frac{1}{1 + \exp(-u_i^T \cdot u_j)}
\]

\[
\hat{p}_1(v_i, v_j) = \frac{w_{ij}}{\sum_{(i', j')} w_{i'j'}}
\]

• Objective:

\[
O_1 = d(\hat{p}_1(\cdot, \cdot), p_1(\cdot, \cdot))
\]

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

\(\bar{u}_i\): Embedding of vertex \(v_i\)

KL-divergence

From Jian Tang’s slides
Preserving the Second-order Proximity

• Given a **directed** edge \((v_i, v_j)\), the conditional probability of \(v_j\) given \(v_i\) is:

\[
p_2(v_j|v_i) = \frac{\exp(\tilde{u}_i^T \cdot \tilde{u}_j)}{\sum_{k=1}^{|V|} \exp(\tilde{u}_k^T \cdot \tilde{u}_i)}
\]

\[
\hat{p}_2(v_j|v_i) = \frac{w_{ij}}{\sum_{k \in V} w_{ik}}
\]

• **Objective:**

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

\[
\alpha - \sum_{(i,j) \in E} w_{ij} \log p_2(v_j|v_i)
\]

\(\tilde{u}_i\): Embedding of vertex \(i\) when \(i\) is a source node; \(\tilde{u}_i'\): Embedding of vertex \(i\) when \(i\) is a target node.

\(\lambda_i\): Prestige of vertex in the network

\(\lambda_i = \sum_j w_{ij}\)

From Jian Tang’s slides
Preserving both Proximity

- Concatenate the embeddings individually learned by the two proximity

From Jian Tang’s slides
Network Embedding Models

• DeepWalk
• Node2vec
• GENE
• LINE

• **SDNE**  (Wang et al., KDD 2016)
SDNE

• Preliminary
  – Autoencoder

\[
\phi : \mathcal{X} \rightarrow \mathcal{F} \\
\psi : \mathcal{F} \rightarrow \mathcal{X} \\
\arg \min_{\phi, \psi} \| X - (\psi \circ \phi) X \|^2
\]
SDNE

• Preliminary
  – Autoencoder
    • The simplest case: a single hidden layer

\[ z = \sigma_1(Wx + b) \]
\[ x' = \sigma_2(W'z + b') \]
\[ \mathcal{L}(x, x') = \|x - x'\|^2 \]
SDNE

- Preliminary
  - Autoencoder
    - The simplest case: a single hidden layer

\[
\begin{align*}
  z &= \sigma_1(Wx + b) \\
  x' &= \sigma_2(W'z + b') \\
  \mathcal{L}(x, x') &= \|x - x'\|^2
\end{align*}
\]
SDNE

- First-order proximity
  - Linked nodes should be coded similarly

\[ L_{1st} = \sum_{i,j=1}^{n} s_{i,j} \| y_i^{(K)} - y_j^{(K)} \|_2^2 \]

\[ = \sum_{i,j=1}^{n} s_{i,j} \| y_i - y_j \|_2^2 \]

\[
\begin{align*}
\mathbf{y}_i^{(1)} &= \sigma(\mathbf{W}^{(1)} \mathbf{x}_i + \mathbf{b}^{(1)}) \\
\mathbf{y}_i^{(k)} &= \sigma(\mathbf{W}^{(k)} \mathbf{y}_i^{(k-1)} + \mathbf{b}^{(k)}), \quad k = 2, \ldots, K
\end{align*}
\]
SDNE

- Second-order proximity
  - The model should reconstruct the neighborhood vectors
  - Similar nodes even without links can have similar codes
    - Or we can not reconstruct the neighborhood

\[ \mathcal{L}_{2nd} = \sum_{i=1}^{n} \left\| (\hat{x}_i - x_i) \odot b_i \right\|_2^2 \]
\[ = \left\| (\hat{X} - X) \odot B \right\|_F^2 \]

\[
\begin{align*}
\mathbf{y}_i^{(1)} &= \sigma(W^{(1)}x_i + b^{(1)}) \\
\mathbf{y}_i^{(k)} &= \sigma(W^{(k)}\mathbf{y}_i^{(k-1)} + b^{(k)}), k = 2, ..., K
\end{align*}
\]
SDNE

• Network reconstruction

<table>
<thead>
<tr>
<th>Method</th>
<th>ARXIV-GRQC</th>
<th>BLOGCATALOG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SDNE</td>
<td>GraRep</td>
</tr>
<tr>
<td>MAP</td>
<td>0.836**</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Significantly outperforms GraRep at the: ** 0.01 level.

• Link prediction

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>P@2</th>
<th>P@10</th>
<th>P@100</th>
<th>P@200</th>
<th>P@300</th>
<th>P@500</th>
<th>P@800</th>
<th>P@1000</th>
<th>P@10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDNE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.99**</td>
<td>0.97**</td>
<td>0.91**</td>
</tr>
<tr>
<td>LINE</td>
<td>1</td>
<td>0.8</td>
<td>0.6</td>
<td>0.655</td>
<td>0.443</td>
<td>0.346</td>
<td>0.2988</td>
<td>0.293</td>
<td>0.1591</td>
</tr>
<tr>
<td>DeepWalk</td>
<td>1</td>
<td>0.2</td>
<td>0.04</td>
<td>0.035</td>
<td>0.033</td>
<td>0.038</td>
<td>0.035</td>
<td>0.035</td>
<td>0.019</td>
</tr>
<tr>
<td>GraRep</td>
<td>1</td>
<td>1</td>
<td>0.96</td>
<td>0.8667</td>
<td>0.968</td>
<td>0.98</td>
<td>0.8775</td>
<td>0.798</td>
<td>0.192</td>
</tr>
<tr>
<td>Common Neighbor</td>
<td>1</td>
<td>1</td>
<td>0.93</td>
<td>0.855</td>
<td>0.827</td>
<td>0.66</td>
<td>0.468</td>
<td>0.391</td>
<td>0.05</td>
</tr>
<tr>
<td>LE</td>
<td>1</td>
<td>1</td>
<td>0.93</td>
<td>0.855</td>
<td>0.827</td>
<td>0.66</td>
<td>0.468</td>
<td>0.391</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Significantly outperforms Line at the: ** 0.01 and * 0.05 level, paired t-test.
Network Embedding Models

- **DeepWalk**
  - Node sentences + word2vec
- **Node2vec**
  - DeepWalk + more sampling strategies
- **GENE**
  - Group~document + doc2vec(DM, DBOW)
- **LINE**
  - Shallow + first-order + second-order proximity
- **SDNE**
  - Deep + First-order + second-order proximity
Applications of Network Embedding

• Basic applications
• Data Visualization
• Text classification
• Recommendation
Basic Applications

• Network reconstruction
• Link prediction
• Clustering
• Feature coding
  – Node classification
    • Demographic prediction
Applications of Network Embedding

• Basic applications
• **Data Visualization** (Tang et al., WWW 2016)
• Text classification
• Recommendation
Figure 1: A typical pipeline of data visualization by first constructing a K-nearest neighbor graph and then projecting the graph into a low-dimensional space.
Data Visualization

• Construction of the KNN graph

For the weights of the edges in the K-nearest neighbor graph, we use the same approach as t-SNE. The conditional probability from data $\vec{x}_i$ to $\vec{x}_j$ is first calculated as:

$$p_{j|i} = \frac{\exp(-||\vec{x}_i - \vec{x}_j||^2/2\sigma_i^2)}{\sum_{(i,k) \in E} \exp(-||\vec{x}_i - \vec{x}_k||^2/2\sigma_i^2)}, \quad \text{and}$$

$$p_{i|i} = 0. \quad (1)$$

Then the graph is symmetrized through setting the weight between $\vec{x}_i$ and $\vec{x}_j$ as:

$$w_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}. \quad (2)$$
Data Visualization

• Visualization-based embedding

\[ P(e_{ij} = 1) = f(||\tilde{y}_i - \tilde{y}_j||), \]

\[ P(e_{ij} = w_{ij}) = P(e_{ij} = 1)^{w_{ij}}. \]

\[ O = \prod_{(i,j)\in E} p(e_{ij} = 1)^{w_{ij}} \prod_{(i,j)\in \bar{E}} (1 - p(e_{ij} = 1))^\gamma \]

\[ \propto \sum_{(i,j)\in E} w_{ij} \log p(e_{ij} = 1) + \sum_{(i,j)\in \bar{E}} \gamma \log(1 - p(e_{ij} = 1)), \]
Data Visualization

- Non-linear function

\[ P(e_{ij} = 1) = f(||\vec{y}_i - \vec{y}_j||), \]

\begin{figure}[h]
\centering
\begin{subfigure}[b]{0.45\textwidth}
\centering
\includegraphics[width=\textwidth]{wikidoc}
\caption{WikiDoc}
\end{subfigure} \hspace{1cm}
\begin{subfigure}[b]{0.45\textwidth}
\centering
\includegraphics[width=\textwidth]{livejournal}
\caption{LiveJournal}
\end{subfigure}
\caption{Comparing different probabilistic functions.}
\end{figure}

\[ f(x) = \frac{1}{1+x^2}. \]
Data Visualization

• **Accuracy**

![Accuracy for 20NG, MNIST, WikiDoc, LiveJournal](images)

• **Running time**

Table 2: Comparison of running time (hours) in graph visualization between the t-SNE and LargeVis.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>20NG</th>
<th>MNIST</th>
<th>WikiWord</th>
<th>WikiDoc</th>
<th>LiveJournal</th>
<th>CSAuthor</th>
<th>DBLPPaper</th>
</tr>
</thead>
<tbody>
<tr>
<td>t-SNE</td>
<td>0.12</td>
<td>0.41</td>
<td>9.82</td>
<td>45.01</td>
<td>70.35</td>
<td>28.33</td>
<td>18.73</td>
</tr>
<tr>
<td>LargeVis</td>
<td>0.14</td>
<td>0.23</td>
<td>2.01</td>
<td>5.60</td>
<td>9.26</td>
<td>4.24</td>
<td>3.19</td>
</tr>
<tr>
<td>Speedup Rate</td>
<td>0</td>
<td>0.7</td>
<td>3.9</td>
<td>7</td>
<td>6.6</td>
<td>5.7</td>
<td>4.9</td>
</tr>
</tbody>
</table>
Data Visualization

(a) 20NG (t-SNE)

(b) 20NG (LargeVis)
Applications of Network Embedding

• Basic applications
• Data Visualization
• **Text classification** (Tang et al., KDD 2015)
• Recommendation
Network embedding helps text modeling

Text representation, e.g., word and document representation, ...
Deep learning has been attracting increasing attention ...
A future direction of deep learning is to integrate unlabeled data ...
The Skip-gram model is quite effective and efficient ...
Information networks encode the relationships between the data objects ...

If we have the word network, we can a network embedding model to learn word representations.

From Jian Tang’s slides
Text Classification

- Adapt the advantages of unsupervised text embedding approaches but naturally utilize the *labeled* data for specific tasks

- Different levels of word co-occurrences: *local context-level, document-level, label-level*

From Jian Tang’s slides
Bipartite Network Embedding

– Extend previous work LINE (Tang et al. WWW’2015) on large-scale information network embedding
  – Preserve the first-order and second-order proximity
  – Only consider the second-order proximity here

• For each edge \((v_i, v_j)\), define a conditional probability

\[
p(v_j | v_i) = \frac{\exp(\mathbf{u}_i^T \cdot \mathbf{u}_j)}{\sum_{j' \in B} \exp(\mathbf{u}_{j'}^T \cdot \mathbf{u}_i)}
\]

• Objective:

\[
O = - \sum_{(i,j) \in E} w_{ij} \log p(v_j | v_i)
\]

• Edge sampling and negative sampling for optimization

Tang et al. **LINE: Large-scale Information Network Embedding.** WWW’2015

From Jian Tang’s slides
Text Classification

Heterogeneous Text Network Embedding

- Heterogeneous text network: three bipartite networks
  - Word-word (word-context), word-document, word-label network
  - Jointly embed the three bipartite networks

- Objective

\[ O_{pte} = O_{ww} + O_{wd} + O_{wl} \]

\[ O_{ww} = - \sum_{(i,j) \in E_{ww}} w_{ij} \log p(v_i | v_j) \]

\[ O_{wd} = - \sum_{(i,j) \in E_{wd}} w_{ij} \log p(v_i | d_j) \]

\[ O_{wl} = - \sum_{(i,j) \in E_{wl}} w_{ij} \log p(v_i | l_j) \]

- where

From Jian Tang’s slides
# Text Classification

## Results on Long Documents: Predictive

<table>
<thead>
<tr>
<th>Type</th>
<th>Algorithm</th>
<th>20newsgroup</th>
<th>Wikipedia</th>
<th>IMDB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Micro-F1</td>
<td>Macro-F1</td>
<td>Micro-F1</td>
</tr>
<tr>
<td>Unsupervised</td>
<td>LINE($G_{wd}$)</td>
<td>79.73</td>
<td>78.40</td>
<td>80.14</td>
</tr>
<tr>
<td>Predictive</td>
<td>CNN</td>
<td>78.85</td>
<td>78.29</td>
<td>79.72</td>
</tr>
<tr>
<td></td>
<td>CNN(pretrain)</td>
<td>80.15</td>
<td>79.43</td>
<td>79.25</td>
</tr>
<tr>
<td></td>
<td>PTE($G_{wl}$)</td>
<td>82.70</td>
<td>81.97</td>
<td>79.00</td>
</tr>
<tr>
<td></td>
<td>PTE($G_{ww} + G_{wl}$)</td>
<td>83.90</td>
<td>83.11</td>
<td>81.65</td>
</tr>
<tr>
<td></td>
<td>PTE($G_{wd} + G_{wl}$)</td>
<td><strong>84.39</strong></td>
<td><strong>83.64</strong></td>
<td>82.29</td>
</tr>
<tr>
<td></td>
<td>PTE(pretrain)</td>
<td>82.86</td>
<td>82.12</td>
<td>79.18</td>
</tr>
<tr>
<td></td>
<td>PTE(joint)</td>
<td><strong>84.20</strong></td>
<td><strong>83.39</strong></td>
<td><strong>82.51</strong></td>
</tr>
</tbody>
</table>

- PTE(joint) > PTE(pretrain)
- PTE(joint) > PTE($G_{wl}$)
- PTE(joint) > CNN/CNN(pretrain)

From Jian Tang’s slides
### Text Classification

Results on **Short Documents: Predictive**

<table>
<thead>
<tr>
<th>Type</th>
<th>Algorithm</th>
<th>DBLP</th>
<th>MR</th>
<th>Twitter</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unsupervised embedding</strong></td>
<td>LINE ($G_{ww} + G_{wd}$)</td>
<td>74.22</td>
<td>70.12</td>
<td>71.13</td>
</tr>
<tr>
<td></td>
<td>CNN</td>
<td>76.16</td>
<td>73.08</td>
<td>72.71</td>
</tr>
<tr>
<td></td>
<td>CNN (pretrain)</td>
<td>75.39</td>
<td>72.28</td>
<td>68.96</td>
</tr>
<tr>
<td><strong>Predictive embedding</strong></td>
<td>PTE($G_{wl}$)</td>
<td>76.45</td>
<td>72.74</td>
<td>73.44</td>
</tr>
<tr>
<td></td>
<td>PTE($G_{ww} + G_{wl}$)</td>
<td>76.80</td>
<td>73.28</td>
<td>72.93</td>
</tr>
<tr>
<td></td>
<td>PTE($G_{wd} + G_{wl}$)</td>
<td><strong>77.46</strong></td>
<td><strong>74.03</strong></td>
<td>73.13</td>
</tr>
<tr>
<td></td>
<td>PTE (pretrain)</td>
<td>76.53</td>
<td>72.94</td>
<td>73.27</td>
</tr>
<tr>
<td></td>
<td>PTE (joint)</td>
<td><strong>77.15</strong></td>
<td>73.61</td>
<td><strong>73.58</strong></td>
</tr>
</tbody>
</table>

**Inferences:**

- PTE(joint) > PTE(pretrain)
- PTE(joint) > PTE($G_{wl}$)
- PTE(joint) ≈ CNN/CNN(pretrain)

From Jian Tang’s slides
Applications of Network Embedding

• Basic applications
• Data Visualization
• Text classification
• **Recommendation** (Zhao et al., AIRO 2016)
Recommendation

• Learning Distributed Representations for Recommender Systems with a Network Embedding Approach
  – Motivation

(a) User-item bipartite net- (b) User-item-tag tripartite work.
Recommendation

• From training records to networks

**Definition 3. Bipartite User-Item (UI) Network.** Let $\mathcal{U}$ denote the set of all the users, and $\mathcal{I}$ denote the set of all the items. A bipartite user-item network can be denoted by $\mathcal{G}^{(bi)} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where the vertex set $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$, the edge set $\mathcal{E} \subset \mathcal{U} \times \mathcal{I}$, the weight matrix $\mathbf{W}$ stores the edge weights, and $W_{u,i}$ denote the link weight between a user $u$ and an item $i$.

**Definition 4. Tripartite User-Item-Tag (UIT) Network.** Let $\mathcal{U}$ denote the set of all the users, $\mathcal{I}$ denote the set of all the items, and $\mathcal{T}$ denote the set of all the tags. A tripartite user-item-tag network can be denoted by $\mathcal{G}^{(tri)} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where the vertex set $\mathcal{V} = \mathcal{U} \cup \mathcal{I} \cup \mathcal{T}$, the edge set $\mathcal{E} \subset ((\mathcal{U} \times \mathcal{I}) \cup (\mathcal{U} \times \mathcal{T}) \cup (\mathcal{I} \times \mathcal{T}))$, and the weight matrix $\mathbf{W}$ stores the edge weights.
Recommendation

• Given any edge in the network

\[
P(e_s, e_t) = \sigma(\mathbf{v}_{es}^\top \cdot \mathbf{v}_{et}) = \frac{1}{1 + \exp(-\mathbf{v}_{es}^\top \cdot \mathbf{v}_{et})}.
\]

\[
\hat{P}(e_s, e_t) = \frac{W_{es,e_t}}{\sum_{(e_{s'}, e_{t'}) \in \mathcal{E}} W_{e_{s'}, e_{t'}}}.
\]

\[
L(\mathcal{G}) = D_{KL}(\hat{P}(\cdot, \cdot) || P(\cdot, \cdot)) \propto \sum_{(e_s, e_t) \in \mathcal{E}} W_{es,e_t} \log P(e_s, e_t).
\]
Recommendation

- User-item recommendation

Table 2. Performance comparisons of the proposed method and baselines on item recommendation.

<table>
<thead>
<tr>
<th>Methods</th>
<th>JD</th>
<th>MovieLens</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P@10</td>
<td>R@10</td>
<td>MAP</td>
</tr>
<tr>
<td>BPR</td>
<td>0.171</td>
<td>0.360</td>
<td>0.337</td>
</tr>
<tr>
<td>DeepWalk</td>
<td>0.259</td>
<td>0.443</td>
<td>0.502</td>
</tr>
<tr>
<td>NERM</td>
<td><strong>0.275</strong></td>
<td><strong>0.477</strong></td>
<td><strong>0.528</strong></td>
</tr>
</tbody>
</table>
Recommendation

• User-item-tag recommendation

Table 4. Performance comparisons of the proposed methods and baselines on tag recommendation.

| Methods  | Last.fm | | | | | | | | Bookmarks | | | | | |
|----------|---------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|          | P@1    | R@1 | F@1 | P@5 | R@5 | F@5 | P@1 | R@1 | F@1 | P@5 | R@5 | F@5 |
| PITF     | 0.305  | 0.125 | 0.178 | 0.189 | 0.351 | 0.245 | 0.381 | 0.132 | 0.197 | 0.204 | 0.304 | 0.244 |
| DeepWalk | 0.088  | 0.044 | 0.059 | 0.040 | 0.099 | 0.057 | 0.064 | 0.024 | 0.035 | 0.038 | 0.074 | 0.050 |
| NERM     | 0.327  | 0.165 | 0.220 | 0.182 | 0.370 | 0.244 | 0.396 | 0.135 | 0.201 | 0.228 | 0.323 | 0.267 |
Conclusions

• There are no boundaries between data types and research areas in terms of mythologies
  – Data models are the core
• Even if the ideas are similar, we can move from shallow to deep if the performance actually improves
Disclaimer

• For convenience, I directly copy some original slides or figures from the referred papers. I am sorry but I did not ask for the permission of each referred author. I thank you for these slides. I will not distribute your original slides.
References

• Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. NIPS 2013: 3111-3119
• Bryan Perozzi, Rami Al-Rfou', Steven Skiena. DeepWalk: online learning of social representations. KDD 2014: 701-710
• Aditya Grover, Jure Leskovec. node2vec: Scalable Feature Learning for Networks. KDD 2016: 855-864
• Jifan Chen, Qi Zhang, Xuanjing Huang, Incorporate Group Information to Enhance Network Embedding. CIKM 2016
• Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, Qiaozhu Mei. LINE: Large-scale Information Network Embedding. WWW 2015: 1067-1077
• Jian Tang, Jingzhou Liu, Ming Zhang, Qiaozhu Mei. Visualizing Large-scale and High-dimensional Data. WWW 2016: 287-297
• Jian Tang, Meng Qu, Qiaozhu Mei. PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks. KDD 2015: 1165-1174
Advanced Readings

• Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C. Aggarwal, Thomas S. Huang: Heterogeneous Network Embedding via Deep Architectures. KDD 2015: 119-128

• Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, Wenwu Zhu: Asymmetric Transitivity Preserving Graph Embedding. KDD 2016: 1105-1114

• Thomas N. Kipf, Max Welling: Semi-Supervised Classification with Graph Convolutional Networks. CoRR abs/1609.02907 (2016)

• Mikael Henaff, Joan Bruna, Yann LeCun: Deep Convolutional Networks on Graph-Structured Data. CoRR abs/1506.05163 (2015)