
VARIANT

• We can differentiate:
  ‣ Node embedding
  ‣ Edge Embedding
  ‣ Substructure embedding
  ‣ Whole graph Embedding

• In this course, only node embedding (often called graph embedding)
• Representation learning on networks
  ‣ **Representation learning = feature learning**, as opposed to manual feature engineering (heuristics)

• Embedding => Latent space
IN CONCRETE TERMS

• A graph is composed of
  ‣ Nodes (possibly with labels)
  ‣ Edges (possibly directed, weighted, with labels)

• A graph/node embedding technique in \( d \) dimensions will assign a vector of length \( d \) to each node, that will be useful for *what we want to do with the graph*.
  ‣ It captures some aspect of the network structure

• A vector can be assigned to an edge \((u,v)\) by combining vectors of \( u \) and \( v \)
WHAT TO DO WITH EMBEDDINGS?

• Two possible ways to use an embedding:
  ‣ Unsupervised learning:
    ‐ The distance between vectors in the embedding is used for *something*
  ‣ Supervised learning:
    ‐ Algorithm learn to predict *something* from the features in the embedding
WHAT CAN WE DO WITH EMBEDDINGS?
EMBEDDING TASKS

• Common tasks:
  ‣ Link prediction (supervised)
  ‣ Graph reconstruction (unsupervised link prediction ? / ad hoc)
  ‣ Community detection (unsupervised)
  ‣ Node classification (supervised community detection ?)
  ‣ Role definition (Variant of node classification, can be unsupervised)
  ‣ Visualisation (distances, like unsupervised)
OVERVIEW OF MOST POPULAR METHODS
PRE-DEEPWALK

MATRIX DECOMPOSITION
LE: LAPLACIAN EIGENMAPS

• Introduced 2001

• Objective function:

\[ y^* = \min \sum_{i \neq j} ||y_i - y_j||^2 S_{ij} \]

- \( y^* \): optimal embedding
- \( y_i \): embedding of node \( i \)
- \( S_{ij} \): similarity between nodes \( i \) and \( j \) (A, heuristic, …)

• Minimize the product between distance in the embedding and similarity in the graph

  › If nodes are similar, they must be close in the embedding
LE: LAPLACIAN EIGENMAPS

- \( y^* = \min \sum_{i \neq j} \| y_i - y_j \|^2 S_{ij} \)

  - Solution: \( d \) eigenvectors of lowest eigenvalues of \( D^{-1/2}LD^{-1/2} \)
  - \( L \): Laplacian, with \( S = A \)

HOPE: HIGHER-ORDER PROXIMITY PRESERVED EMBEDDING

- Preserve a proximity matrix

\[ y^* = \min \sum_{i,j} |S_{ij} - y_i y_j^T | \]

- \( S \) can be the adjacency matrix, or number of common neighbors, Adamic Adar, etc.

- As similarity tends towards 0, embedding vectors must tend towards orthogonality (orthogonal vectors: \( y_i y_j^T = 0 \))

LLE: LOCALLY LINEAR EMBEDDING

• Introduced 2000

• A node features can be represented as a linear combination of its neighbors' 

\[ y_i = \sum_j A_{ij} y_j \]

• Objective function:

\[ y^* = \min \sum_i \|y_i - \sum_j A_{ij} y_j\|^2 \]

RANDOM WALKS BASED
DEEPWALK

• The first Random Walk+Neural Networks graph embedding method.
  ‣ First of a long series

• Adaptation of word2vec/skipgram to graphs

SKIPGRAM

Word embedding
Corpus => Word = vectors
Similar embedding= similar context

Source Text

The quick brown fox jumps over the lazy dog. →

Training Samples

(the, quick)
(the, brown)
(quick, the)
(quick, brown)
(quick, fox)
(brown, the)
(brown, quick)
(brown, fox)
(brown, jumps)

fox, quick)
(fox, brown)
(fox, jumps)
(fox, over)

[http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/]
Skipgram

Output weights for “car”

Word vector for “ants” \times \text{softmax} = \text{Probability that if you randomly pick a word nearby “ants”, that it is “car”}

https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b
SKIPGRAM

\[ N = \text{embedding size. } V = \text{vocabulary size} \]

https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b
SKIPGRAM

[https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/]
Table 8: Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Example 1</th>
<th>Example 2</th>
<th>Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>France - Paris</td>
<td>Italy: Rome</td>
<td>Japan: Tokyo</td>
<td>Florida: Tallahassee</td>
</tr>
<tr>
<td>big - bigger</td>
<td>small: larger</td>
<td>cold: colder</td>
<td>quick: quicker</td>
</tr>
<tr>
<td>Miami - Florida</td>
<td>Baltimore: Maryland</td>
<td>Dallas: Texas</td>
<td>Kona: Hawaii</td>
</tr>
<tr>
<td>Einstein - scientist</td>
<td>Messi: midfielder</td>
<td>Mozart: violinist</td>
<td>Picasso: painter</td>
</tr>
<tr>
<td>Sarkozy - France</td>
<td>Berlusconi: Italy</td>
<td>Merkel: Germany</td>
<td>Koizumi: Japan</td>
</tr>
<tr>
<td>copper - Cu</td>
<td>zinc: Zn</td>
<td>gold: Au</td>
<td>uranium: plutonium</td>
</tr>
<tr>
<td>Berlusconi - Silvio</td>
<td>Sarkozy: Nicolas</td>
<td>Putin: Medvedev</td>
<td>Obama: Barack</td>
</tr>
<tr>
<td>Microsoft - Windows</td>
<td>Google: Android</td>
<td>IBM: Linux</td>
<td>Apple: iPhone</td>
</tr>
<tr>
<td>Microsoft - Ballmer</td>
<td>Google: Yahoo</td>
<td>IBM: McNealy</td>
<td>Apple: Jobs</td>
</tr>
<tr>
<td>Japan - sushi</td>
<td>Germany: bratwurst</td>
<td>France: tapas</td>
<td>USA: pizza</td>
</tr>
</tbody>
</table>
GENERIC “SKIPGRAM”

- Algorithm that takes an input:
  - The element to embed
  - A list of “context” elements

- Provide as output:
  - An embedding with interesting properties
    - Works well for machine learning
    - Similar elements are close in the embedding
    - Somewhat preserves the overall structure
DEEPWALK

• Skipgram for graphs:
  ‣ 1) Generate “sentences” using random walks
  ‣ 2) Apply Skipgram

• Parameters:
  ‣ Embedding dimensions $d$
  ‣ Context size
  ‣ More technical parameters: length of random walks, number of walks starting from each node, etc.

NODE2VEC

• Use biased random walk to tune the context to capture *what we want*
  ‣ “Breadth first” like RW => local neighborhood (edge probability ?)
  ‣ “Depth-first” like RW => global structure ? (Communities ?)
  ‣ 2 parameters to tune:
    - \( p \): bias towards revisiting the previous node
    - \( q \): bias towards exploring undiscovered parts of the network

Figure 2: Illustration of the random walk procedure in node2vec. The walk just transitioned from \( t \) to \( v \) and is now evaluating its next step out of node \( v \). Edge labels indicate search biases \( \alpha \).
RANDOM WALK METHODS

• What is the objective function?

• How to interpret the distance between nodes in the embedding?
RANDOM WALK METHODS

Approximately

\[ y = \min \sum_{(i,j)} p(n_j | n_i) - \sigma(y_i y_j^T) \]

with \( p(w_j | w_i) \) the probability to encounter node \( n_j \) in a random walk of a chosen length starting from node \( n_i \). Its objective is therefore to make the distance in the embedding proportional to a random walk based distance in the graph.

with \( \sigma \) the softmax function defined as \( \frac{e^x}{\sum e^x} \), a function commonly used in neural networks to add non-linearity and to ensure that the solution is a probability.

RANDOM WALK METHODS

• Scalability:
  ‣ Skipgram uses techniques from machine learning developed for very large datasets: highly scalable (not necessarily fast or cost efficient)

• Matrix factorization methods require the similarity matrix $S$ as input
  ‣ Computing all random walk distance: $\mathcal{O}(n^2)$
  ‣ $k$ random walks of length $\ell$ from each node: $\mathcal{O}(n)$
Minimize a global loss defined as:

\[
L = \sum_{(v_i, v_j) \in E} \ell(DEC(z_i, z_j), s_G(v_i, v_j))
\]

**DEC**: Decoder function (e.g., \(DEC(z_i, z_j) = z_i^T z_j\))

**s_G**: Ground truth similarity (e.g., \(s_G(v_i, v_j) = A_{ij}\))

**\(\ell\)**: Chosen loss function (e.g., \(\ell(a, b) = |a - b|\))

ENCODER DECODER FRAMEWORK

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Decoder</th>
<th>Proximity measure</th>
<th>Loss function (ℓ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix factorization</td>
<td>Laplacian Eigenmaps [4]</td>
<td></td>
<td></td>
<td>z_i - z_j</td>
</tr>
<tr>
<td></td>
<td>Graph Factorization [1]</td>
<td>z_i^T z_j</td>
<td>A_i,j</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GraRep [9]</td>
<td>z_i^T z_j</td>
<td>A_i,j, A_i,j, ..., A_i,j</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HOPE [44]</td>
<td>z_i^T z_j</td>
<td>A_i,j</td>
<td></td>
</tr>
<tr>
<td>Random walk</td>
<td>DeepWalk [46]</td>
<td>\frac{e_i^T z_j}{\sum_k \in V e_i^T z_k}</td>
<td>p_G(v_j</td>
<td>v_i)</td>
</tr>
<tr>
<td></td>
<td>node2vec [27]</td>
<td>\frac{e_i^T z_j}{\sum_k \in V e_i^T z_k}</td>
<td>p_G(v_j</td>
<td>v_i) (biased)</td>
</tr>
</tbody>
</table>

\( p_G(v_j | v_i) \): probability of visiting \( v_j \) on a fixed-length random walk started from \( v_i \)
SOME REMARKS ON WHAT ARE EMBEDDINGS
An adjacency matrix is an “embedding”… in high dimension

That represents the structural equivalence
- 2 nodes have similar “embeddings” if they have similar neighborhoods
- Distance=>$\#$ of different neighbors (Manhattan Distance)

Standard dimensionality reduction (T-SNE, PCA) of this matrix?
- Small dimensions
- But still unintuitive notion of distance
Graph layouts are also embeddings.
  ‣ Force layout, kamada-kawai ....

They try to put connected nodes close to each other and non-connected ones “not close”

Problem: they usually try to avoid overlaps

Often not scalable
NODE EMBEDDING: VISUALIZATION
FROM D TO 2

• Graph embedding can be used to visualize graphs

• Requires to reduce the embedding from d to 2
  ‣ TSNE
  ‣ PCA
  ‣ ...

• Interpretable positions of nodes

• But not necessarily optimized for human reading
CLIQUE RING

5 cliques of size 20 with 1 edge between them

Spring layout

LLE

LE

n2v
NODE EMBEDDING: COMMUNITY DETECTION
CLUSTERING EMBEDDINGS

- Many algorithm exists for clustering non-network data
  - K-means, DBscan, etc.

- Clustering: group nodes that are close in the feature space.
EMBEDDING ROLES
STRUC2VEC/ROLE2VEC

• In node2vec/Deepwalk, the context collected by RW contains the **labels** of encountered nodes

• Instead, we could memorize the **properties** of the nodes: attributes if available, or computed attributes (degrees, CC, …)

• =>Nodes with a same context will be nodes in a same “position” in the graph

• =>Capture the role of nodes instead of proximity

NODE CLASSIFICATION WITH EMBEDDINGS
NODE CLASSIFICATION

• To each node is associated a vector in the embedding
  ‣ This vector corresponds to topological features of the node, used instead of, for instance, centralities
  ‣ Both types of features can be combined

• As usual, a classifier can be trained using those features
## NODE CLASSIFICATION

### Table: Performance Comparison

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>BlogCatalog</th>
<th>PPI</th>
<th>Wikipedia</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spectral Clustering</td>
<td>0.0405</td>
<td>0.0681</td>
<td>0.0395</td>
</tr>
<tr>
<td>DeepWalk</td>
<td>0.2110</td>
<td>0.1768</td>
<td>0.1274</td>
</tr>
<tr>
<td>LINE</td>
<td>0.0784</td>
<td>0.1447</td>
<td>0.1164</td>
</tr>
<tr>
<td>node2vec</td>
<td><strong>0.2581</strong></td>
<td><strong>0.1791</strong></td>
<td><strong>0.1552</strong></td>
</tr>
<tr>
<td>node2vec settings (p,q)</td>
<td>0.25, 0.25</td>
<td>4, 1</td>
<td>4, 0.5</td>
</tr>
<tr>
<td>Gain of node2vec [%]</td>
<td><strong>22.3</strong></td>
<td><strong>1.3</strong></td>
<td><strong>21.8</strong></td>
</tr>
</tbody>
</table>

Controversies...

---

LINK PREDICTION WITH EMBEDDINGS

UNSUPERVISED LINK PREDICTION

- Unsupervised link prediction \textbf{from embeddings}
- $\Rightarrow$ Compute the distance between nodes in the embedding
- $\Rightarrow$ Use it as a similarity score
SUPERVISED LINK PREDICTION

• Supervised link prediction from embeddings

• =>embeddings provide features for nodes (nb features: dimensions)
  ‣ Combine nodes features to obtain edge features

• =>Train a classifier to predict edges based on features from the embedding
SUPERVISED LINK PREDICTION

Combining nodes vectors into edge vectors

<table>
<thead>
<tr>
<th>Operator</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>$(a + b)/2$</td>
</tr>
<tr>
<td>Concat</td>
<td>$[a_1, \ldots, a_d, b_1, \ldots, b_d]$</td>
</tr>
<tr>
<td>Hadamard</td>
<td>$[a_1 \times b_1, \ldots, a_d \times b_d]$</td>
</tr>
<tr>
<td>Weighted L1</td>
<td>$[</td>
</tr>
<tr>
<td>Weighted L2</td>
<td>$[(a_1 - b_1)^2, \ldots, (a_d - b_d)^2]$</td>
</tr>
</tbody>
</table>
SUPERVISED LINK PREDICTION

- How well does it work?
- According to creators articles
  - Node2vec (2016)
  - VERSE (2018)
- => These methods are better than the state of the art

### TABLE 4: AREA UNDER CURVE (AUC) SCORES FOR LINK PREDICTION

<table>
<thead>
<tr>
<th>Op</th>
<th>Algorithm</th>
<th>Dataset</th>
<th>arXiv</th>
<th>PPI</th>
<th>Facebook</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Common Neighbors</td>
<td>0.8100</td>
<td>0.7142</td>
<td>0.8153</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Jaccard’s Coefficient</td>
<td>0.8880</td>
<td>0.7018</td>
<td>0.8067</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Adamic-Adar</td>
<td>0.8289</td>
<td>0.7126</td>
<td>0.8315</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pref. Attachment</td>
<td>0.7137</td>
<td>0.6670</td>
<td>0.6996</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Spectral Clustering</td>
<td>0.5960</td>
<td>0.6588</td>
<td>0.5812</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DeepWalk</td>
<td>0.7238</td>
<td>0.6923</td>
<td>0.7066</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LINE</td>
<td>0.7029</td>
<td>0.6330</td>
<td>0.6516</td>
<td></td>
</tr>
<tr>
<td>node2vec</td>
<td></td>
<td>0.7266</td>
<td>0.7543</td>
<td>0.7221</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Spectral Clustering</td>
<td>0.6192</td>
<td>0.4920</td>
<td>0.5740</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DeepWalk</td>
<td><strong>0.9680</strong></td>
<td>0.7441</td>
<td>0.9340</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LINE</td>
<td>0.9490</td>
<td>0.7249</td>
<td>0.8902</td>
<td></td>
</tr>
<tr>
<td>node2vec</td>
<td></td>
<td><strong>0.9680</strong></td>
<td><strong>0.7719</strong></td>
<td><strong>0.9366</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Spectral Clustering</td>
<td>0.7200</td>
<td>0.6356</td>
<td>0.7099</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DeepWalk</td>
<td>0.9574</td>
<td>0.6026</td>
<td>0.8282</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LINE</td>
<td>0.9483</td>
<td>0.7024</td>
<td>0.8809</td>
<td></td>
</tr>
<tr>
<td>node2vec</td>
<td></td>
<td>0.9602</td>
<td>0.6292</td>
<td>0.8468</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Spectral Clustering</td>
<td>0.7107</td>
<td>0.6026</td>
<td>0.6765</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DeepWalk</td>
<td>0.9584</td>
<td>0.6118</td>
<td>0.8305</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LINE</td>
<td>0.9460</td>
<td>0.7106</td>
<td>0.8862</td>
<td></td>
</tr>
<tr>
<td>node2vec</td>
<td></td>
<td>0.9606</td>
<td>0.6236</td>
<td>0.8477</td>
<td></td>
</tr>
</tbody>
</table>

(a) Average, (b) Hadamard, (c) Weighted-L1, and (d) Weighted-L2 (AUC)
LINK PREDICTION

- Personal opinion: not that simple

LINK PREDICTION

• First few predictions: advantage to heuristics

Better prediction at distance 2, worst otherwise


(a) FACEBOOK

(b) ASTROPH
MODEL STACKING
FOR LINK PREDICTION
Table S12. Average AUC, precision, and recall performances of the link prediction algorithms over 124 social networks as a subset of CommunityFitNet corpus. A random forest is used for supervised stacking of methods. Here, the predictors are adjusted for maximum F measure using a model selection through a cross validation on training set. The results are reported on 20% holdout test set.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>AUC</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>0.89 ± 0.07</td>
<td>0.42 ± 0.13</td>
<td>0.85 ± 0.08</td>
</tr>
<tr>
<td>Q-MR</td>
<td>0.87 ± 0.07</td>
<td>0.38 ± 0.16</td>
<td>0.78 ± 0.07</td>
</tr>
<tr>
<td>Q-MP</td>
<td>0.86 ± 0.08</td>
<td>0.25 ± 0.07</td>
<td>0.83 ± 0.09</td>
</tr>
<tr>
<td>B-NR (SBM)</td>
<td>0.93 ± 0.06</td>
<td>0.3 ± 0.08</td>
<td>0.85 ± 0.12</td>
</tr>
<tr>
<td>B-NR (DC-SBM)</td>
<td>0.93 ± 0.07</td>
<td>0.28 ± 0.08</td>
<td>0.88 ± 0.08</td>
</tr>
<tr>
<td>cICL-HKK</td>
<td>0.93 ± 0.08</td>
<td>0.34 ± 0.1</td>
<td>0.85 ± 0.14</td>
</tr>
<tr>
<td>B-HKK</td>
<td>0.88 ± 0.07</td>
<td>0.17 ± 0.05</td>
<td>0.79 ± 0.17</td>
</tr>
<tr>
<td>Infomap</td>
<td>0.91 ± 0.04</td>
<td>0.29 ± 0.08</td>
<td>0.83 ± 0.05</td>
</tr>
<tr>
<td>MDL (SBM)</td>
<td>0.94 ± 0.07</td>
<td>0.31 ± 0.09</td>
<td>0.87 ± 0.16</td>
</tr>
<tr>
<td>MDL (DC-SBM)</td>
<td>0.93 ± 0.09</td>
<td>0.26 ± 0.09</td>
<td>0.89 ± 0.11</td>
</tr>
<tr>
<td>S-NB</td>
<td>0.94 ± 0.07</td>
<td>0.3 ± 0.1</td>
<td>0.87 ± 0.08</td>
</tr>
<tr>
<td>mean model-based</td>
<td>0.91 ± 0.08</td>
<td>0.3 ± 0.12</td>
<td>0.84 ± 0.12</td>
</tr>
<tr>
<td>mean indiv. topol.</td>
<td>0.64 ± 0.19</td>
<td>0.2 ± 0.27</td>
<td>0.56 ± 0.33</td>
</tr>
<tr>
<td>mean indiv. topol. &amp; model</td>
<td>0.7 ± 0.21</td>
<td>0.22 ± 0.25</td>
<td>0.62 ± 0.32</td>
</tr>
<tr>
<td>emb-DW</td>
<td>0.95 ± 0.1</td>
<td>0.45 ± 0.16</td>
<td>0.92 ± 0.13</td>
</tr>
<tr>
<td>emb-vgae</td>
<td>0.95 ± 0.08</td>
<td>0.09 ± 0.02</td>
<td>0.96 ± 0.09</td>
</tr>
<tr>
<td>all topol.</td>
<td>0.97 ± 0.08</td>
<td>0.89 ± 0.21</td>
<td>0.88 ± 0.2</td>
</tr>
<tr>
<td>all model-based</td>
<td>0.95 ± 0.07</td>
<td>0.76 ± 0.2</td>
<td>0.88 ± 0.17</td>
</tr>
<tr>
<td>all embed.</td>
<td>0.95 ± 0.11</td>
<td>0.75 ± 0.23</td>
<td>0.74 ± 0.23</td>
</tr>
<tr>
<td>all topol. &amp; model</td>
<td>0.98 ± 0.06</td>
<td>0.89 ± 0.22</td>
<td>0.88 ± 0.19</td>
</tr>
<tr>
<td>all topol. &amp; embed.</td>
<td>0.96 ± 0.1</td>
<td>0.86 ± 0.22</td>
<td>0.83 ± 0.25</td>
</tr>
<tr>
<td>all model &amp; embed.</td>
<td>0.96 ± 0.09</td>
<td>0.78 ± 0.21</td>
<td>0.74 ± 0.22</td>
</tr>
<tr>
<td>all topol., model &amp; embed.</td>
<td>0.97 ± 0.09</td>
<td>0.86 ± 0.23</td>
<td>0.84 ± 0.23</td>
</tr>
</tbody>
</table>