Deep Learning for Network Biology

Marinka Zitnik and Jure Leskovec
Stanford University
This Tutorial

snap.stanford.edu/deepnetbio-ismb

ISMB 2018

July 6, 2018, 2:00 pm - 6:00 pm
This Tutorial

1) Node embeddings
   - Map nodes to low-dimensional embeddings
   - Applications: PPIs, Disease pathways

2) Graph neural networks
   - Deep learning approaches for graphs
   - Applications: Gene functions

3) Heterogeneous networks
   - Embedding heterogeneous networks
   - Applications: Human tissues, Drug side effects
Part 1: Node Embeddings

Some materials adapted from:
• Hamilton et al. 2018. Representation Learning on Networks. WWW.
Embedding Nodes

Intuition: Map nodes to d-dimensional embeddings such that similar nodes in the graph are embedded close together.
Setup

- Assume we have a graph $G$:
  - $V$ is the vertex set
  - $A$ is the adjacency matrix (assume binary)

- No node features or extra information is used!
Goal: Map nodes so that similarity in the embedding space (e.g., dot product) approximates similarity in the network.
Embedding Nodes

Goal: $\text{similarity}(u, v) \approx z_v^T z_u$

Need to define!

Input network

$\text{ENC}(u)$

$\text{ENC}(v)$

d-dimensional embedding space

encode nodes

$z_u$

$z_v$
1. Define an encoder (a function $ENC$ that maps node $u$ to embedding $z_u$)

2. Define a node similarity function (a measure of similarity in the input network)

3. Optimize parameters of the encoder so that:

$$\text{similarity}(u, v) \approx z_v^\top z_u$$
Two Key Components

1. **Encoder** maps a node to a \(d\)-dimensional vector:

\[
ENC(v) = z_v
\]

- node in the input graph
- \(d\)-dimensional embedding

2. **Similarity function** defines how relationships in the input network map to relationships in the embedding space:

\[
similarity(u, v) \approx z_v^\top z_u
\]

- Similarity of \(u\) and \(v\) in the network
- Dot product between node embeddings
Embedding Methods

- Many methods use similar encoders:
  - node2vec, DeepWalk, LINE, struc2vec

- These methods use different notions of node similarity:
  - Two nodes have similar embeddings if:
    - they are connected?
    - they share many neighbors?
    - they have similar local network structure?
    - etc.
Outline of This Section

1. Adjacency-based similarity
2. Random walk approaches
3. Biomedical applications
Adjacency-based Similarity

Material based on:
• Ahmed et al. 2013. Distributed Natural Large Scale Graph Factorization. WWW.
Adjacency-based Similarity

- **Similarity function** is the edge weight between $u$ and $v$ in the network
- **Intuition:** Dot products between node embeddings approximate edge existence

\[
\mathcal{L} = \sum_{(u,v) \in V \times V} \left( \frac{z_u^T z_v}{\|A_{u,v}\|^2} - A_{u,v}\right)^2
\]
Adjacency-based Similarity

\[ \mathcal{L} = \sum_{(u,v) \in V \times V} \| z_u^\top z_v - A_{u,v} \|^2 \]

- Find embedding matrix \( Z \in \mathbb{R}^{d \times |V|} \) that minimizes the loss \( \mathcal{L} \):
  - Option 1: Stochastic gradient descent (SGD)
    - Highly scalable, general approach
  - Option 2: Solve matrix decomposition solvers
    - e.g., SVD or QR decompositions
    - Need to derive specialized solvers
Adjacency-based Similarity

- $O(|V|^2)$ runtime
  - Must consider all node pairs
  - $O(|E|)$ if summing over non-zero edges (e.g., Natarajan et al., 2014)

- $O(|V|)$ parameters
  - One learned embedding per node

- Only consider direct connections

Red nodes are obviously more similar to Green nodes compared to Orange nodes, despite none being directly connected
Outline of This Section

1. Adjacency-based similarity ✔
2. Random walk approaches
3. Biomedical applications
Random Walk Approaches

Material based on:
• Perozzi et al. 2014. DeepWalk: Online Learning of Social Representations. KDD.
• Grover et al. 2016. node2vec: Scalable Feature Learning for Networks. KDD.
• Ribeiro et al. 2017. struc2vec: Learning Node Representations from Structural Identity. KDD.
Idea: Define node similarity function based on higher-order neighborhoods

- **Red:** Target node
- **k=1:** 1-hop neighbors
  - $A$ (i.e., adjacency matrix)
- **k=2:** 2-hop neighbors
- **k=3:** 3-hop neighbors

How to stochastically define these higher-order neighborhoods?
Unsupervised Feature Learning

- **Intuition:** Find embedding of nodes to $d$-dimensions that preserves similarity

- **Idea:** Learn node embedding such that nearby nodes are close together

- **Given a node $u$, how do we define nearby nodes?**
  - $N_R(u)$ … neighbourhood of $u$ obtained by some strategy $R$
Feature Learning as Optimization

- Given $G = (V, E)$
- Goal is to learn $f: u \rightarrow \mathbb{R}^d$
  - where $f$ is a table lookup
    - We directly “learn” coordinates $z_u = f(u)$ of $u$
- Given node $u$, we want to learn feature representation $f(u)$ that is predictive of nodes in $u$’s neighborhood $N_R(u)$

$$\max_f \sum_{u \in V} \log \Pr(N_R(u) | z_u)$$
Unsupervised Feature Learning

Goal: Find embedding $z_u$ that predicts nearby nodes $N_R(u)$:

$$\sum_{v \in V} \log(P(N_R(u) | z_u))$$

Assume conditional likelihood factorizes:

$$P(N_R(u) | z_u) = \prod_{n_i \in N_R(u)} P(n_i | z_u)$$
Random-walk Embeddings

\[
\mathbf{Z}_u^\top \mathbf{Z}_v \approx \text{Probability that } u \text{ and } v \text{ co-occur in a random walk over the network}
\]
Why Random Walks?

1. **Flexibility**: Stochastic definition of node similarity:
   - Local and higher-order neighborhoods

2. **Efficiency**: Do not need to consider all node pairs when training
   - Consider only node pairs that co-occur in random walks
Random Walk Optimization

1. Simulate many short random walks starting from each node using a strategy $R$

2. For each node $u$, get $N_R(u)$ as a sequence of nodes visited by random walks starting at $u$

3. For each node $u$, learn its embedding by predicting which nodes are in $N_R(u)$:

$$
\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} - \log(P(v | z_u))
$$
Random Walk Optimization

\[
\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} \text{sum over all nodes } u \quad \text{sum over nodes } v \quad \text{predicted probability of } u \text{ and } v \text{ co-occurring on random walks starting from } u
\]

\[
- \log \left( \frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)} \right) \quad \text{predicted probability of } u \text{ and } v \text{ co-occurring on random walk, i.e., use softmax to parameterize } P(v|z_u)
\]

Random walk embeddings = \( z_u \) minimizing \( \mathcal{L} \)
Random Walk Optimization

But doing this naively is too expensive!

\[
\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} - \log \left( \frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)} \right)
\]

Nested sum over nodes gives \(O(|V|^2)\) complexity!

The problem is normalization term in the softmax function?
Solution: Negative sampling \((\text{Mikolov et al., 2013})\)

\[
\log \left( \frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)} \right)
\]

\[
\approx \log(\sigma(z_u^T z_v)) - \sum_{i=1}^{k} \log(\sigma(z_u^T z_{n_i})), n_i \sim P_V
\]

i.e., instead of normalizing w.r.t. all nodes, just normalize against \(k\) random negative samples
Random Walks: Overview

1. Simulate many short random walks starting from each node using a strategy $R$
2. For each node $u$, get $N_R(u)$ as a sequence of nodes visited by random walks starting at $u$
3. For each node $u$, learn its embedding by predicting which nodes are in $N_R(u)$:

$$
L = \sum_{u \in V} \sum_{v \in N_R(u)} - \log(P(v | z_u))
$$

Can efficiently approximate using negative sampling
What is the strategy $R$?

- **So far:**
  - Given simulated random walks, we described how to optimize node embeddings

- **What strategies can we use to obtain these random walks?**
  - Simplest idea:
    - Fixed-length, unbiased random walks starting from each node (i.e., DeepWalk from Perozzi et al., 2013)
  - **Can we do better?**
    - Grover et al., 2016; Ribeiro et al., 2017; Abu-El-Haija et al., 2017 and many others
node2vec: Biased Walks

**Idea:** Use flexible, biased random walks that can trade off between local and global views of the network (Grover and Leskovec, 2016)

---

![Diagram of node2vec with BFS and DFS search strategies](Image)
node2vec: Biased Walks

Two classic strategies to define a neighborhood $N_R(u)$ of a given node $u$:

$N_{BFS}(u) = \{ s_1, s_2, s_3 \}$ \hspace{2cm} Local microscopic view

$N_{DFS}(u) = \{ s_4, s_5, s_6 \}$ \hspace{2cm} Global macroscopic view
Interpolate BFS and DFS

Biased random walk $R$ that given a node $u$ generates neighborhood $N_R(u)$

- Two parameters:
  - Return parameter $p$:
    - Return back to the previous node
  - In-out parameter $q$:
    - Moving outwards (DFS) vs. inwards (BFS)
Biased Random Walks

Biased 2\textsuperscript{nd}-order random walks explore network neighborhoods:

- Rnd. walk started at $u$ and is now at $w$
- **Insight:** Neighbors of $w$ can only be:
  - Closer to $u$
  - Same distance to $u$
  - Farther from $u$

**Idea:** Remember where that walk came from
Biased Random Walks

- Walker is at $w$. Where to go next?

- $p$, $q$ model transition probabilities
  - $p$ ... return parameter
  - $q$ ... "walk away" parameter

$1/p, 1/q, 1$ are unnormalized probabilities
Biased Random Walks

- Walker is at \( w \). Where to go next?

- **BFS-like** walk: Low value of \( p \)
- **DFS-like** walk: Low value of \( q \)

\[ N_S(u) \] are the nodes visited by the walker

\[
\begin{align*}
&\begin{array}{c}
\text{w} \\
\text{s}_1 \\
\text{s}_2 \\
\text{s}_3 \\
\text{u}
\end{array} \\
&\begin{array}{c}
1/p \\
1/q \\
1/p \\
1/q \\
1/p \\
1/q
\end{array}
\]

Unnormalized transition prob.
BFS vs. DFS

BFS:
Micro-view of neighbourhood

DFS:
Macro-view of neighbourhood
Experiment: Micro vs. Macro

Interactions of characters in a novel:

\[
\begin{align*}
P = 1, \quad q = 2 & \quad \text{Microscopic view of the network neighbourhood} \\
p = 1, \quad q = 0.5 & \quad \text{Macroscopic view of the network neighbourhood}
\end{align*}
\]
**Summary So Far**

- **Idea:** Embed nodes so that distances in the embedding space reflect node similarities in the network.

- **Different notions of node similarity:**
  - Adjacency-based (i.e., similar if connected)
  - Random walk approaches:
    - Fixed-length, unbiased random walks starting from each node in the original network (Perozzi et al., 2013)
    - Fixed-length, biased random walks on the original network (node2vec, Grover et al., 2016)
Summary So Far

- **So what method should I use..?**
- No one method wins in all cases….
  - e.g., node2vec performs better on node classification while multi-hop methods performs better on link prediction ([Goyal and Ferrara, 2017 survey](https://snap.stanford.edu/deepnetbio-ismb)).
- Random walk approaches are generally more efficient (i.e., $O(|E|)$ vs. $O(|V|^2)$)
- **In general:** Must choose def’n of node similarity that matches application!
Outline of This Section

1. Adjacency-based similarity
2. Random walk approaches
3. Biomedical applications
Biomedical Applications

Material based on:

- Agrawal et al. 2018. [Large-scale analysis of disease pathways in the human interactome](https://pubs.american chemical society.org/doi/10.1021/acs.pcs.7b00396). *PSB.*
Biomedical Applications

1. **Disease pathway detection:**
   - Identify proteins whose mutation is linked with a particular disease
   - **Task:** Multi-label node classification

2. **Protein interaction prediction:**
   - Identify protein pairs that physically interact in a cell
   - **Task:** Link prediction
Human Interactome

- RAD50
- MSH4
- MSH5
- PCNA
- RAD51
- BRCA2
- FEN1
- RFC1
- MED6
- DMC1
Human Interactome

Key principle (Cowen et al., 2017): Proteins that interact underlie similar phenotypes (e.g., diseases)
Disease Pathways

- **Pathway**: Subnetwork of interacting proteins associated with a disease

![Diagram of Disease Pathways](image)

- MSH4
- MSH5
- RAD50
- PCNA
- RAD51
- BRCA2
- RFC1
- MED6
- DMC1
- FEN1

Lung carcinoma pathway
Disease Pathways: Task

- Known (seed) disease protein
- Predicted disease protein
- Predicted protein-disease association

Disease protein discovery

- Protein
- Disease protein
- Protein-protein interaction
- Protein-disease association
- Pathway component
Disease Pathway Dataset

- Protein-protein interaction (PPI) network culled from 15 knowledge databases:
  - 350k physical interactions, e.g., metabolic enzyme-coupled interactions, signaling interactions, protein complexes
  - All protein-coding human genes (21k)
- Protein-disease associations:
  - 21k associations split among 519 diseases
- Multi-label node classification: every node (i.e., protein) can have 0, 1 or more labels (i.e., disease associations)
Experimental Setup

- Two main stages:
  1. Take the PPI network and use node2vec to learn an embedding for every node
  2. For each disease:
     - Fits a logistic regression classifier that predicts disease proteins based on the embeddings:
       - Train the classifier using training proteins
       - Predict disease proteins in the test test: probability that a particular protein is associated with the disease
Pathways: Results

- **Best performers:**
  - node2vec embeddings
    - hits@100 = 0.40
  - DIAMOnD
    - hits@100 = 0.30
  - Matrix completion
    - hits@100 = 0.29

- **Worst performer:**
  - Neighbor scoring
    - hits@100 = 0.24

hits@100: fraction of all the disease proteins are ranked within the first 100 predicted proteins
1. **Disease pathway detection:**
   - Identify proteins whose mutation is linked with a particular disease
   - **Task:** Multi-label node classification

2. **Protein interaction prediction:**
   - Identify protein pairs that physically interact in a cell
   - **Task:** Link prediction
Protein-Protein Interaction

Network Data

- Human PPI network:
  - Experimentally validated physical protein-protein interactions from the BioGRID

- **Link prediction:** Given two proteins, predict probability that they interact
Learning Edge Embeddings

- **So far:** Methods learn embeddings for nodes:
  - Great for tasks involving individual nodes (e.g., node classification)

- **Question:** How to address tasks involving pairs of nodes (e.g., link prediction)?

- **Idea:** Given $u$ and $v$, define an operator $g$ that generates an embedding for pair $(u, v)$:

  $$z_{(u,v)} = g(u, v)$$
How to define operator $g$?

- **Desiderata:** The operator needs to be defined for any pair of nodes, even if the nodes are not connected.

- We consider four choices for $g$:

<table>
<thead>
<tr>
<th>Scoring node pairs</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Average</td>
<td>$[z_u \oplus z_v]_i = \frac{z_u(i) + z_v(i)}{2}$</td>
</tr>
<tr>
<td>(b) Hadamard</td>
<td>$[z_u \odot z_v]_i = z_u(i) \times z_v(i)$</td>
</tr>
<tr>
<td>(c) Weighted-L1</td>
<td>$|z_u \cdot z_v|_{1i} =</td>
</tr>
<tr>
<td>(d) Weighted-L2</td>
<td>$|z_u \cdot z_v|_{2i} =</td>
</tr>
</tbody>
</table>
Experimental Setup

- We are given a PPI network with a certain fraction of edges removed:
  - Remove about 50% of edges
  - Randomly sample an equal number of node pairs at random which have no edge connecting them
  - Explicitly removed edges and non-existent (or false) edges form a balanced test data set

- Two main stages:
  1. Use node2vec to learn an embedding for every node in the filtered PPI network
  2. Predict a score for every protein pair in the test set based on the embeddings
PPI Prediction: Results

Learned embeddings drastically outperform heuristic scores

Hadamard operator:
- Highly stable
- Best average performance

F1 – scores are in [0,1], higher is better

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

Scoring node pairs

- (a) Average
- (b) Hadamard
- (c) Weighted-L1
- (d) Weighted-L2

<table>
<thead>
<tr>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[z_u \boxplus z_v]_i = \frac{z_u(i) + z_v(i)}{2}$</td>
</tr>
<tr>
<td>$[z_u \boxdot z_v]_i = z_u(i) * z_v(i)$</td>
</tr>
<tr>
<td>$|z_u \cdot z_v|_1 =</td>
</tr>
<tr>
<td>$|z_u \cdot z_v|_2 = (z_u(i) - z_v(i))^2$</td>
</tr>
</tbody>
</table>
Biomedical Applications

1. Disease pathway detection:
   - Identify proteins whose mutation is linked with a particular disease
   - **Task:** Multi-label node classification

2. Protein interaction prediction:
   - Identify protein pairs that physically interact in a cell
   - **Task:** Link prediction
Outline of This Section

1. Adjacency-based similarity
2. Random walk approaches
3. Biomedical applications
PhD Students

Claire Donnat
Mitchell Gordon
David Hallac
Emma Pierson
Geet Sethi
Himabindu Lakkaraju
Rex Ying
Tim Althoff
Will Hamilton
Alex Porter

Post-Doctoral Fellows

Baharan Mirzasoleiman
Marinka Zitnik
Michele Catasta
Srijan Kumar

Research Staff

Stephen Bach
Adrijan Bradaschia
Rok Sosic

Industry Partnerships

Funding

Collaborators

Dan Jurafsky, Linguistics, Stanford University
Christian Danescu-Miculescu-Mizil, Information Science, Cornell University
Stephen Boyd, Electrical Engineering, Stanford University
David Gleich, Computer Science, Purdue University
VS Subrahmanian, Computer Science, University of Maryland
Sarah Kunz, Medicine, Harvard University
Russ Altman, Medicine, Stanford University
Jochen Profit, Medicine, Stanford University
Eric Horvitz, Microsoft Research
Jon Kleinberg, Computer Science, Cornell University
Sendhil Mullainathan, Economics, Harvard University
Scott Delp, Bioengineering, Stanford University
Jens Ludwig, Harris Public Policy, University of Chicago
Many interesting high-impact projects in Machine Learning and Large Biomedical Data

Applications: Precision Medicine & Health, Drug Repurposing, Drug Side Effect modeling, Network Biology, and many more