Part 1: Next-Gen Machine Learning for Network Biology

Marinka Zitnik
Stanford University
Two Lectures

Part 1: May 15, 2019, 2:30 pm - 4:00 pm
- **Methodology:** Shallow network embeddings:
  - Map nodes to low-dimensional features
- **Resources:** Data, tools, codebases
- **Applications:** PPIs, Disease pathways, Tissues

Part 2: May 16, 2019, 9:00 am – 10:30 am
- **Methodology:** Deep network embeddings:
  - Graph neural networks for rich biomedical graphs
- **Resources:** Data, practical advice and demos
- **Applications:** Polypharmacy, Drug repurposing
1. Used new methods to predict safety, side effects of drug combinations:
   - First-ever systematic and predictive study of drug combinations
   - Follow-up research on prostate cancer and validations in the clinic

2. Used new methods to repurpose old drugs for new diseases:
   - Outperforms baselines by up to 172%
   - Correctly predicted drugs repurposed at Stanford

Interactive feedback loops for AI
Two Lectures

Part 1: May 15, 2019, 2:30 pm - 4:00 pm
- Methodology: Shallow network embeddings:
  - Map nodes to low-dimensional features
- Resources: Data, tools, codebases
- Applications: PPIs, Disease pathways, Tissues

Part 2: May 16, 2019, 9:00 am – 10:30 am
- Methodology: Deep network embeddings:
  - Graph neural networks for rich biomedical graphs
- Resources: Data, practical advice and demos
- Applications: Polypharmacy, Drug repurposing
Science crucially depends on scientific instruments

Physical instruments facilitate discoveries

Microscope

Robert Hooke, *Micrographia*, 1665

Need instruments for modern, data intensive sciences

Knowledge discovery
However: Biomedical data present challenges for knowledge discovery

**Multi-scale:** molecules, individuals, populations

**Heterogeneous:** experimental readouts, curated annotations, metadata

**Confounded:** data from different labs, biotech platforms, organisms

Biomedical problems

Complex, multi-scale, heterogeneous datasets

Machine learning

Tabular, monolithic, flat matrix-like datasets

Significant gap between what ML can address and real-world biomedical problems

To close the gap one has to:
1. Develop a general mathematical representation to integrate heterogeneous data in their broadest sense
2. Develop methods for learning over such representation to open doors for new discoveries
Outline of this Lecture

1) Biological Networks
   - Why networks? Why is learning on networks hard?

2) Node embeddings
   - Methodology: Map nodes to vector representations
   - Applications: PPIs, Disease pathways

3) Heterogeneous networks
   - Methodology: Embedding heterogeneous networks
   - Applications: Human tissues
Networks allow for integration of biomedical data

Multiple scales

Heterogeneity
Why Networks? Why Now?

- **Question:** How are human genetic diseases and the corresponding disease genes related to each other?
- **Findings:** Genes associated with similar diseases are likely to interact and have similar expression.

**Why Networks? Why Now?**

- **Question:** How to simulate a basic eukaryotic cell?
- **Findings:** Simulations reveal molecular mechanisms of cell growth, drug resistance and synthetic life.

![Diagram showing network interactions in a cell](image-url)

Why Networks? Why Now?

- **Question:** How to discover heterogeneity of cancer?
- **Findings:** Analysis identifies new cancer subtypes with distinct patient survival


Patient subtype

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>115</td>
<td>115</td>
</tr>
</tbody>
</table>
Why Networks? Why Now?

- **Question:** How to study ecological systems?
- **Findings:** Pollinators interact with flowers in one season but not in another, and the same flower species interact with both pollinators and herbivores.

Why Networks? Why Now?

- **Question:** What are features of human microbiome?

- **Findings:** Microbiota reflects the seasonal availability of different types of food and differentiate industrialized and traditional populations.

Many Data are Networks

Patient networks

Hierarchies of cell systems

Disease pathways

Genetic interaction networks

Gene co-expression networks

Cell-cell similarity networks
How to do machine learning on biomedical networks?

Networks are a powerful data representation, but are challenging to work with for prevailing deep models.

Predictions, e.g., properties of cells, patient outcomes, new relationships like disease-gene associations, new functional modules.
Prevailing Deep Models

Primarily designed for grids or simple sequences:

- CNNs for fixed-size images/grids
- RNNs for text/sequences

These models brought extraordinary gains in computer vision, natural language processing, speech, and robotics.

But are unable to consider interactions, the essence of biomedical networks.
Biomedical Networks

Real-world networks look like this:

or this:

Examples:

Human contact networks, Disease networks, Patient networks, Cell similarity networks, Medical knowledge graphs
Why is deep learning on networks hard?

Biomedical networks are far more complex!

- Complex topographical structure (no spatial locality like grids)
- No fixed node ordering/reference point (isomorphism problem)
- Different types of entities (nodes) and interactions (edges)
- Rich and heterogeneous features about entities and interactions

Need methods that generalize convolutions beyond simple lattices and learn and reason over rich networks.
Outline of this Lecture

1) Biological networks
   - Why networks? Why is learning on networks hard?

2) Node embeddings
   - Methodology: Map nodes to vector representations
   - Applications: PPIs, Disease pathways

3) Heterogeneous networks
   - Methodology: Embedding heterogeneous networks
   - Applications: Human tissues
Part 2: Node Embeddings

Based on material from:
Objective: Map nodes to $d$-dimensional embeddings such that nodes with similar network neighborhoods are embedded close together.

How to learn mapping function $f$?
Example: Disease Similarity Network

Next: How to learn mapping function $f$?
Assume we have a graph $G$:

- $V$ is the vertex set
- $A$ is the adjacency matrix (binary):
  - Weighted, typed and dynamic graphs as well as multi-graphs (see next part & Thursday’s lecture)
  - Integration of node/edge features, and extra information (see next part & Thursday’s lecture)
**Embedding Nodes**

**Goal:** Map nodes so that similarity in the embedding space (e.g., dot product) approximates similarity in the network.

- **Input network**
- **d-dimensional embedding space**
- **Encode nodes**
- \( ENC(u) \)
- \( ENC(v) \)
- \( Z_u \)
- \( Z_v \)
Goal:

\[ \text{similarity}(u, v) \approx z_v^T z_u \]

Input network

Encode nodes

d-dimensional embedding space
Learning Node Embeddings

1. Define an encoder (a function $\text{ENC}$ that maps node $u$ to embedding $z_u$)

2. Define a node similarity function (measure of similarity in the 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:
   $$\text{ENC}(v) = z_v$$
   node in the input graph

2. **Similarity function** defines how relationships in the input network map to relationships in the embedding space:
   $$\text{similarity}(u, v) \approx z_v^\top z_u$$
   dot product between node embeddings

Similarity of $u$ and $v$ in the network
Embedding Methods

- Many methods use similar encoders:
  - matrix factorizations, node2vec, DeepWalk, LINE, struc2vec
- These methods use different notions of node similarity:
  - Two nodes have similar embeddings if:
    - they are connected (i.e., matrix factorization)?
    - they share many neighbors?
    - they have similar local network structure?
    - etc.
Outline of This Section

1. Shallow node embeddings
2. Biomedical applications
Shallow Node Embeddings

Based on material from:
Node Similarity

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?
Node Embeddings

- **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 $R$
Optimization Task

- 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 $N_R(u)$

$$\max_f \sum_{u \in V} \log \Pr(N_R(u) \mid z_u)$$
Optimization Task

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:

$$P(N_R(u)|z_u) = \prod_{n_i \in N_R(u)} P(n_i|z_u)$$
Node Similarity Function Based on Random Walks

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

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

- Sum over all nodes \( u \)
- Sum over nodes \( v \) seen on random walks starting from \( u \)
- Predicted probability of \( u \) and \( v \) 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!

\[ \mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} - \log \left( \frac{\exp(\mathbf{z}_u \mathbf{z}_v)}{\sum_{n \in V} \exp(\mathbf{z}_u^\top \mathbf{z}_n)} \right) \]

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

The problem is normalization term in the softmax function?
Solution: Negative Sampling

Solution: Negative sampling (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

sigmoid function

random distribution over all nodes
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 $N_R(u)$:

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

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 to use to get 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?**
    - Node2vec (Grover et al., 2016)
    - Struc2vec (Ribeiro et al., 2017)
    - Abu-El-Haija et al., 2017 and many others
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 \hspace{0.5cm} microscopic \hspace{0.5cm} view

$N_{DFS}(u) = \{ s_4, s_5, s_6 \}$  \hspace{2cm} Global \hspace{0.5cm} macroscopic \hspace{0.5cm} view
Experiment: Local vs. Global

Local view of network (Homophily)

Global view of network (Structural similarity)
struc2vec: Structural Similarity

- **Goal:** Nodes visited by random walks starting from node \( u \) should be structurally similar to \( u \):
  - E.g., \( u \) and \( v \) are structurally similar, have similar local network structure.

\[ u \text{ and } v \text{ are far apart in the network but are structurally similar!} \]
struc2vec: Three Main Steps

1. Compute structural similarity of nodes based on k-hop neighborhoods

2. Construct a new multilayer graph:
   - K-th layer measures structural similarity of nodes w.r.t. k-hop neighborhoods

3. Run weighted random walks on the multilayer graph to generate $N_R(u)$
struc2vec: Step 1

Let $N_k(u)$ be nodes in $k$-hop neighborhood of $u$

**Lemma:** $u$ and $v$ are structurally equivalent considering $k$-hop neighborhoods:

- $G[N_k(u)]$ and $G[N_k(v)]$ are isomorphic graphs
- $N_{k-1}(u)$ and $N_{k-1}(v)$ have identical ordered degree sequence

**Idea:** Get structural similarity of $u$ and $v$ by looking at their ordered degree sequences $N_{k-1}(u)$ and $N_{k-1}(v)$
Construct a multilayer graph:

- All nodes from the original network are in every layer.
- **K-th layer**: Structural similarity of nodes w.r.t. k-hop neighborhoods.
- **Edge weights**: Proportional to nodes’ structural similarity.
struc2vec: Step 3

Multilayer graph:
- $K$-th layer has structural similarity of nodes w.r.t. $k$-hop neighborhoods

Idea: Use the multilayer graph to get $N_R(u)$:
- $N_R(u)$ is a sequence of nodes visited by weighted random walk starting at $u$
struc2vec: Overview

1. Construct the multilayer graph and simulate many random walks starting from each node

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 $N_R(u)$:

$$
\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} - \log(P(v | z_u))
$$
struc2vec: Experiment

Barbell graph:

- Ground-truth: Nodes of the same color are structurally equivalent (i.e., their local network structure is the same)
struc2vec: Experiment

Nodes that are structurally similar are embeded close together!
Beyond struc2vec: GraphWave

Node colors indicate structural roles. Not available to the algorithm during training.
Summary so Far

**Approach:** Embed nodes such that:
- Algebraic operations in the learned space reflect topology of the graph

Different notions of **node similarity:**
- Adjacency-based (i.e., similar if connected)
- Multi-hop similarity definitions
- Random walk approaches

**In general:** Must choose define node similarity that matches application!
Outline of This Section

1. Shallow node embeddings

2. Biomedical applications
Based on material from:

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
Key principle (Cowen et al., 2017):
Proteins that interact underlie similar phenotypes (e.g., diseases)
Pathway: Subnetwork of interacting proteins associated with a disease

Disease Pathways

- 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 node embeddings

2. For each disease:
   - Fit a classifier that predicts disease proteins based on the embeddings:
     - Train the classifier using training proteins
     - Predict a probability that a test 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
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
Protein-Protein Interactions

Network Data

- Human PPI network:
  - Experimentally validated physical protein-protein interactions

- **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 \((u, v)\):

\[
  z_{(u,v)} = g(u, v)
\]
Learning Edge Embeddings

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>[ \left[ \mathbf{z}_u \boxplus \mathbf{z}_v \right]_i = \frac{\mathbf{z}_u(i) + \mathbf{z}_v(i)}{2} ]</td>
</tr>
<tr>
<td>(b) Hadamard</td>
<td>[ \left[ \mathbf{z}_u \boxtimes \mathbf{z}_v \right]_i = \mathbf{z}_u(i) \ast \mathbf{z}_v(i) ]</td>
</tr>
<tr>
<td>(c) Weighted-L1</td>
<td>[ \left| \mathbf{z}_u \cdot \mathbf{z}_v \right|_1 \equiv \left</td>
</tr>
<tr>
<td>(d) Weighted-L2</td>
<td>[ \left| \mathbf{z}_u \cdot \mathbf{z}_v \right|_2 \equiv \left</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

- 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

### Table: PPI Prediction Results

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

**Scoring node pairs**

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

**Definition**

- \[ (\mathbf{z}_u \oplus \mathbf{z}_v)_i = \frac{\mathbf{z}_u(i) + \mathbf{z}_v(i)}{2} \]
- \[ (\mathbf{z}_u \odot \mathbf{z}_v)_i = \mathbf{z}_u(i) \cdot \mathbf{z}_v(i) \]
- \[ ||\mathbf{z}_u \cdot \mathbf{z}_v||_1 = ||\mathbf{z}_u(i) - \mathbf{z}_v(i)|| \]
- \[ ||\mathbf{z}_u \cdot \mathbf{z}_v||_2 = ||\mathbf{z}_u(i) - \mathbf{z}_v(i)||^2 \]

F1 – scores are in [0,1], higher is better
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. Random walk approaches

2. Biomedical applications
Outline of this Lecture

1) Biological networks
   - Why networks? Why is learning on networks hard?

2) Node embeddings
   - Methodology: Map nodes to vector representations
   - Applications: PPIs, Disease pathways

3) Heterogeneous networks
   - Methodology: Embedding heterogeneous networks
   - Applications: Human tissues
Part 3: Heterogeneous Networks

Based on material from:

• Zitnik et al., 2017. Predicting multicellular function through multi-layer tissue networks. ISMB & Bioinformatics.
So far we focused on homogeneous networks!

Can we embed heterogeneous networks, i.e., het nets, knowledge graphs?
Many Het Nets in Biology

Diagram showing different modes and link types in biological networks.
Motivating Problem: Prediction of Protein Functions

Proteins are worker molecules

- Biomedical and pharma implications

Functions depend on tissue context

- Proteins in similar tissues share similar features
- Functions in heart are different from functions in the brain, etc.
Why is protein function prediction across tissues hard?

1) Multiscale, hierarchical organization of tissues:
   - Tissues are related to each other
   - Proteins in similar tissues have similar functions

2) Many tissues have no annotations:
   - Need to predict functions in a tissue without any protein functions (node labels) in that tissue

3) Previous research [Radivojac et al.’13, Cho et al.’16; Kramer et al.’14; Yu et al.’15; etc.]
   - Protein functions assumed constant across tissues
     - Functions in heart are the same as in skin
     - Functions in the brain are the same as in skin
Motivating Problem: What Does My Protein Do?

**Goal:** Given a protein, a tissue, and a function, predict how likely the protein has that function in that tissue

Protein × (Function, Tissue) → [0,1]
Multimodal Tissue Networks

Tissue-specific protein interaction networks + tissue hierarchy
Multimodal Tissue Networks

How to learn mapping functions $f_i$?

Input

Output

$u \rightarrow \mathbb{R}^d$
Setup: Multimodal Networks

**Input:** Graphs $\{G_i\}_i$, hierarchy $\mathcal{M}$
- Graphs $\{G_i\}_{i=1..T}$ are in leaves of $\mathcal{M}$

**Goal:** Learn functions: $f_i : V_i \rightarrow \mathbb{R}^d$

**Multi-scale model:**
- Four layers: $i, j, k, l$
- Three scales: “3”, “2”, “1”

**Output:** Node embeddings:
- For each graph $G_i$
- For each sub-hierarchy
Embedding Approach

Two components:

1. **For each graph** $G_i$:
   Embed nodes with similar local topology close together

2. **For hierarchy** $M$:
   Encourage nodes in similar graphs to share similar features
Single-Graph Objective

- **Intuition:** In each graph, embed nodes to \(d\) dimensions
- **Approach:** Nodes \(u\) and \(v\) are similar if their network neighborhoods are similar
- **Given node** \(u\) in graph \(G_i\), neighborhood \(N_i(u)\) is defined based on random walks starting at node \(u\)
Single-Graph Objective

- Given node $u$ in graph $G_i$, learn $u$’s embedding such that it predicts nearby nodes $N_i(u)$:

$$
\omega_i(u) = \log Pr(N_i(u) | f_i(u))
$$

- Given $T$ graphs $\{G_i\}_{i=1..T}$, maximize:

$$
\Omega_i = \sum_{u \in V_i} \omega_i(u), \quad \text{for } i = 1, 2, \ldots, T
$$
Summary so Far

We have not yet considered hierarchy $\mathcal{M}$:

- Node embeddings in different graphs are learned independently of each other

How to model dependencies between graphs when learning node embeddings?
Recall: Hierarchy of Graphs

- Hierarchy $M$ is a tree, given by the parent-child relationships:

$$\pi : M \rightarrow M$$

- $\pi(i)$ is parent of $i$ in $M$

Example:

“2” is parent of $G_i, G_j$
Cross-Graph Objective

For hierarchy $\mathcal{M}$:

- Encode dependencies between graphs $G_i$
- **Recursive regularization:**
  - Embeddings at level $i$ are encouraged to be similar to embeddings in $i$’s parent in the hierarchy.

![Diagram of cross-graph objective](image)
Cross-Graph Objective

- Given \( u \), learn \( u \)'s embedding in \( G_i \) to be close to \( u \)'s embedding in parent \( \pi(i) \):

\[
c_i(u) = \frac{1}{2} \| f_i(u) - f_{\pi(i)}(u) \|_2^2
\]

- Multi-scale: Repeat at every level of \( \mathcal{M} \)

\[
C_i = \sum_{u \in L_i} c_i(u)
\]

\( L_i \) has all graphs appearing in sub-hierarchy rooted at \( i \)
Embedding Approach: Optimization

Solve the maximum likelihood problem:

\[
\max_{f_1, f_2, \ldots, f_{|M|}} \sum_{i \in T} \Omega_i - \lambda \sum_{j \in M} C_j.
\]

Single-graph objective

Cross-graph objective
Embedding Approach: Algorithm

1. For each graph $G_i$:
   - Sample fixed-length random walks starting from each node $u \in G_i$

2. Optimize the objective using stochastic gradient descent

**Scalability:** No pairwise comparison of nodes from different graphs:

$$O \left( \sum_{i,j} |V_i| |V_j| \right) \rightarrow O \left( T \sum_i |V_i| \right)$$
Biomedical Application

Based on material from:


What Does My Protein Do?

**Goal:** Given a protein, a tissue, and a function, predict how likely the protein has that function in that tissue

$\text{Protein} \times (\text{Function, Tissue}) \rightarrow [0,1]$
Data: 107 Tissue Graphs

- **Graphs** are PPI nets:
  - Nodes: proteins
  - Edges: tissue-specific PPIs

- **Tissue hierarchy**:
  - BREENDA tissue ontology

- **Node labels**:
  - Tissue-specific protein functions
  - GIANT / Human Base:
    - E.g., Function Cortex development in renal cortex tissue
    - E.g., Function Artery morphogenesis in artery tissue
Experimental Setup

- **Task:** Multi-label node classification:
  - E.g., Does RPT1 play a role in angiogenesis in blood tissue?

- Every node (protein) is assigned one or more labels (functions)

- **Setup:**
  - Learn features for multimodal network
  - Train a classifier for each function based on a fraction of proteins and all their functions
  - Predict functions for new proteins
Results: Protein Function Prediction Across Tissues

**OhmNet**

- Standard function prediction methods: >10% improvement over function prediction methods
- Homogeneous network embeddings: >18% improvement over non-hierarchical versions of the dataset
- Tensor decomposition: >21% improvement over matrix-based methods

Substantial improvement over methods that ignore tissue-specific information
Case Study: 9 Brain Tissues

9 brain tissue PPI networks in two-level hierarchy
Multi-Scale Node Embeddings

Brainstem

Brain

Learned protein embeddings match human anatomy
Application: Transfer Learning

Predict protein function in a brand new functionally uncharacterized tissue

Task: Predict functions in target tissue without access to any annotation/label in that tissue

<table>
<thead>
<tr>
<th>Target tissue</th>
<th>AUROC (Original)</th>
<th>AUROC (Transfer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Natural killer cell</td>
<td>0.834 (± 0.076)</td>
<td>0.776 (± 0.063)</td>
</tr>
<tr>
<td>Placenta</td>
<td>0.830 (± 0.082)</td>
<td>0.758 (± 0.068)</td>
</tr>
<tr>
<td>Spleen</td>
<td>0.803 (± 0.030)</td>
<td>0.779 (± 0.043)</td>
</tr>
<tr>
<td>Liver</td>
<td>0.803 (± 0.047)</td>
<td>0.741 (± 0.025)</td>
</tr>
<tr>
<td>Forebrain</td>
<td>0.796 (± 0.036)</td>
<td>0.755 (± 0.037)</td>
</tr>
<tr>
<td>Macrophage</td>
<td>0.789 (± 0.037)</td>
<td>0.724 (± 0.024)</td>
</tr>
<tr>
<td>Epidermis</td>
<td>0.785 (± 0.030)</td>
<td>0.749 (± 0.032)</td>
</tr>
<tr>
<td>Hematopoietic stem c.</td>
<td>0.784 (± 0.035)</td>
<td>0.744 (± 0.036)</td>
</tr>
<tr>
<td>Blood plasma</td>
<td>0.784 (± 0.027)</td>
<td>0.703 (± 0.039)</td>
</tr>
<tr>
<td>Smooth muscle</td>
<td>0.778 (± 0.031)</td>
<td>0.729 (± 0.041)</td>
</tr>
<tr>
<td>Average</td>
<td>0.799</td>
<td>0.746</td>
</tr>
</tbody>
</table>

42% improvement over baselines
Outline of this Lecture

1) Biological networks
   - Why networks? Why is learning on networks hard

2) Node embeddings
   - Methodology: Map nodes to vector representations
   - Applications: PPIs, Disease pathways

3) Heterogeneous networks
   - Methodology: Embedding heterogeneous networks
   - Applications: Human tissues
Lecture Resources

- **MAMBO:** Multimodal biomedical networks
  - Scales to networks with 2.3 billion edges and over 2,000 modes
  - [snap.stanford.edu/mambo](http://snap.stanford.edu/mambo)

- **Network data:**
  - [snap.stanford.edu/projects.html](http://snap.stanford.edu/projects.html):
    - CRank, Decagon, MAMBO, NE, OhmNet, Pathways, Tree of Life, and many others
  - [snap.stanford.edu/biodata](http://snap.stanford.edu/biodata)
    - Algorithm benchmarking, method development
    - Easy to link entities across datasets

<table>
<thead>
<tr>
<th>Name</th>
<th>Edges</th>
<th>Entities</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC-Neuron</td>
<td>49,471,006</td>
<td>cell, cell</td>
<td>Similarity network between cells in embryonic mouse brain</td>
</tr>
<tr>
<td>ChCh-Miner</td>
<td>96,137</td>
<td>drug, drug</td>
<td>Interactions between FDA-approved drugs</td>
</tr>
<tr>
<td>ChChSe-Decagon</td>
<td>4,649,441</td>
<td>drug, drug, side-effect</td>
<td>Side effects of drug combinations</td>
</tr>
<tr>
<td>ChG-InterDecagon</td>
<td>131,034</td>
<td>drug, gene</td>
<td>Chemical-gene interaction network</td>
</tr>
</tbody>
</table>
Two Lectures

Part 1: May 15, 2019, 2:30 pm - 4:00 pm

- **Methodology**: Shallow network embeddings:
  - Map nodes to low-dimensional features
- **Resources**: Data, tools, codebases
- **Applications**: PPIs, Disease pathways, Tissues

Part 2: May 16, 2019, 9:00 am – 10:30 am

- **Methodology**: Deep network embeddings:
  - Graph neural networks for rich biomedical graphs
- **Resources**: Data, practical advice and demos
- **Applications**: Polypharmacy, Drug repurposing