Towards Precision Medicine with Graph Representation Learning

Michelle M. Li & Marinka Zitnik
Department of Biomedical Informatics
Broad Institute of Harvard and MIT
Harvard Data Science

zitniklab.hms.harvard.edu/biomedgraphml
Tutorial VT4
July 7, 2022 at 9am – 1pm CDT

All tutorial materials are available at zitniklab.hms.harvard.edu/biomededgraphml
Biology is interconnected

The effects of drugs are not limited to the molecules to which they directly bind in the body. Instead, these effects spread throughout biological networks in which they act. Therefore, the effect of a drug on a disease is inherently a network phenomenon.
Graph representation learning realizes key network principles for data-rich biomedicine

Cellular components associated with a specific disease (phenotype) show a tendency to cluster in the same network neighborhood

Deep graph representation learning methods are well-suited for the analysis of biological networks
This Tutorial

1. **Methods**: Network diffusion, shallow network embeddings, and graph neural networks

2. **Applications**: Fundamental biological discoveries and precision medicine

3. **Outlook**: Future directions and Q&A session

4. **Hands-on exercises**: Demos, implementation details, tools, and tips
Graph representation learning tasks

- **Node-level**
  - Characteristics of a given node

- **Edge-level**
  - Whether or how two nodes are connected

- **Subgraph-level**
  - How clusters of nodes interact with each other and the rest of the graph

- **Graph-level**
  - Similarity of graph to other graphs

---

Graph Representation Learning for Biomedicine, *Nature Biomedical Engineering* (in press), 2022, arXiv:2104.04883

Towards Precision Medicine with Graph Representation Learning - bit.ly/biomedicalgml - ISMB 2022
Graph representation learning tasks

- **Node-level**
  - Characteristics of a given node

Graph representation learning tasks
Node classification: Example

Classifying the function of proteins in the interactome!

Graph representation learning tasks

- **Edge-level**
  - Whether or how two nodes are connected

![Graph representation learning diagram](image)
Link prediction: Example

Predicting which diseases a new molecule might treat!

Drugs

Diseases

“Treats” relationship

Unknown drug-disease relationship

Graph representation learning tasks

- **Subgraph-level**
  - How clusters of nodes interact with each other and the rest of the graph
Subgraph classification: Example

Identifying disease proteins in the interactome!

Graph representation learning tasks

- **Graph-level**
  - Similarity of graph to other graphs

Graph Representation Learning for Biomedicine, *Nature Biomedical Engineering* (in press), 2022, arXiv:2104.04883
Graph classification: Example

Designing new small molecule compounds to treat a disease!

Predictive modeling lifecycle

(Supervised) Machine learning lifecycle: This feature, that feature. Every single time!

Raw Data → Structured Data → Learning Algorithm → Model

Feature Engineering
Automatically learn features

Downstream prediction task
Feature learning in graphs

Goal: Efficient task-independent feature learning for machine learning in networks!

\[ f: u \rightarrow \mathbb{R}^d \]

Feature representation, embedding
Embedding nodes

**Intuition:** Map nodes to embeddings such that similar nodes in graph are embedded closeby.

\[ f(\text{Input}) = \text{Output} \]

*How to learn mapping function* \( f \)?
Predominant graph learning paradigms

a Graph theoretic techniques

b Random walks and diffusion

c Persistent homology

d Geometrical and topological representations

e Manifold learning

f Shallow network embeddings

g Graph neural networks

h Graph generative models

Graph Representation Learning for Biomedicine, *Nature Biomedical Engineering* (in press), 2022, arXiv:2104.04883
Random walks and diffusion

- Nodes in a graph influence each other along paths
- Diffusion measures these spreads of influences

Intuition
- Capture the local connectivity patterns for each node
- Define node similarity function based on higher-order neighborhoods

Red: Target node
k = 1: 1-hop neighbors
k = 2: 2-hop neighbors
k = 3: 3-hop neighbors
Random walks and diffusion

- **Method:** Diffusion state distance (DSD)
  - Simulate random walks from source node $u$
  - Count the number of random walks of length $k$ that start at $u$ and visit a destination node $v$

- **Node representation:** Each node $u$ has vector $\Psi_u$ that represents its influence on its $k$-hop neighborhood

- **Comparison:** Calculate whether nodes $u$ and $v$ have similar local connectivity by $DSD(u, v) = || \Psi_u - \Psi_v ||_1$
Application: Identify disease pathways

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

Large-scale analysis of disease pathways in the human interactome, *Pacific Symposium on Biocomputing, 2018*
Disease pathway dataset

- **Protein-protein interaction (PPI) network** culled from 15 knowledge databases:
  - 350k physical interactions
    - **Examples**: 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)

Large-scale analysis of disease pathways in the human interactome, *Pacific Symposium on Biocomputing, 2018*
Experimental setup

- Two main stages:
  1. Take the PPI network and use DSD to compute a vector representation for every node
  2. For each disease, fit a logistic regression classifier that predicts disease proteins based on the vector representations:
     - Train the classifier using training proteins
     - Predict disease proteins in the test set (i.e. the probability that a particular protein is associated with the disease)
Predominant graph learning paradigms

a Graph theoretic techniques

b Random walks and diffusion

c Persistent homology

d Geometrical and topological representations

e Manifold learning

f Shallow network embeddings

g Graph neural networks

h Graph generative models

Graph Representation Learning for Biomedicine, *Nature Biomedical Engineering* (in press), 2022, arXiv:2104.04883
Shallow network embeddings

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

- **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!

$$f(G) = \text{Disease similarity network} = \text{2-dimensional node embeddings}$$
Shallow network embeddings

- **Goal:** Similarity in the embedding space approximates similarity in the network

- **Three main stages:**
  1. Given a pair of nodes $u, v$ in network $G$, obtain function $f$ to map these nodes to an embedding space to generate $h_u$ and $h_v$
  2. Define network similarity $f_n(u,v)$ and embedding similarity $f_z(h_u, h_v)$
  3. Define loss $l$ to measure whether embedding preserves distance in original graph, and optimize by minimizing the loss

[Graph representation learning for biomedicine, Nature Biomedical Engineering (in press), 2022, arXiv:2104.04883]
Shallow network embeddings

- **Summary:**
  - One-layer of data transformation
  - A single hidden layer maps node $u$ to embedding $h_u$ via function $f$

- **Limitations:**
  - $O(|V|)$ parameters are needed:
    - No sharing of parameters between nodes
    - Every node has its own unique embedding
  - Inherently “transductive”
    - Cannot generate embeddings for nodes not seen during training
  - Do not incorporate node features
    - Many graphs have features that we can and should leverage
Application: Predict protein interactions

- **Human PPI network:**
  - Experimentally validated physical protein-protein interactions (BioGRID)
- **Link prediction:** Given two proteins, predict probability that they interact

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

- Given \( u \) and \( v \), define an operator \( g \) that generates an embedding for pair \((u, v)\):
  \[ h_{(u,v)} = g(u, v) \]

- Examples of 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 _i = \frac{z_{u}(i) + z_{v}(i)}{2}</td>
</tr>
<tr>
<td>(b) Hadamard</td>
<td>[ z_{u} \odot z_{v} ] _i _i = z_{u}(i) _i \times z_{v}(i)</td>
</tr>
<tr>
<td>(c) Weighted-L1</td>
<td>| z_{u} \cdot z_{v} |<em>{1} _i = | z</em>{u}(i) - z_{v}(i) |</td>
</tr>
<tr>
<td>(d) Weighted-L2</td>
<td>| z_{u} \cdot z_{v} |<em>{2} _i = | z</em>{u}(i) - z_{v}(i) |^{2}</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 that have no edge connecting them
  - Explicitly removed edges and non-existent (or false) edges form a balanced test data set

- Two main stages:
  1. 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
Predominant graph learning paradigms

- **a** Graph theoretic techniques
  - Degree: 4
  - Closeness: 0.3
  - Centrality: 0.2
  - Betweenness: 0.1
  - Entropy: 0.3

- **b** Random walks and diffusion
  - Transition matrix: \( \Psi \)
  - Nodes: \( \Psi_u \)

- **c** Persistent homology
  - Persistent diagram:

- **d** Geometrical and topological representations
  - Data: \( D \)
  - Topology: \( T \)
  - Clusters: \( C_1, C_2, C_3 \)

- **e** Manifold learning
  - Dimensionality reduction:
  - \( \text{dim}(\mathbb{R}) >> \text{dim}(\mathbb{E}) \)

- **f** Shallow network embeddings
  - Function: \( f \)
  - Objective function:
  - Minimize \( l(f_z(h_u, h_v), f_n(u, v)) \)

- **g** Graph neural networks
  - Message passing:

- **h** Graph generative models
  - Graph: \( G \)
  - Random walk: \( \tilde{G} \)

Graph Representation Learning for Biomedicine, *Nature Biomedical Engineering* (in press), 2022, arXiv:2104.04883
Graph neural networks

- **Encoder:** Multiple layers of nonlinear transformation of graph structure

Diagram:
- Graph convolutions
- Regularization, e.g., dropout
- Graph convolutions
Convolutional networks

- Let’s start with convolutional networks on an image:

- Single convolutional network with a 3x3 filter:

- Transform information (or messages) from the neighbors and combine them: $\sum_i W_i h_i$
Real world graphs

But what if your graphs look like this?

Gene interaction network  Disease pathways  Biomedical knowledge graphs

Examples:
- Biological or medical networks
- Social networks
- Information networks
- Knowledge graphs
- Communication networks
- Web graphs
- …
Naïve approach

- Join adjacency matrix and features
- Feed them into a deep neural network:

Issues with this idea:
- \( O(N) \) parameters
- Not applicable to graphs of different sizes
- Not invariant to node ordering
Graph neural networks

- **Intuition:**
  - Each node’s neighborhood defines a computational graph
  - Generate node embeddings based on local network neighborhoods

- **Neighborhood aggregation:**

  ![Diagram of graph neural networks](image)

  **Neural networks**

  **Layer 0:**
  - Model can be of arbitrary depth
  - Nodes have embeddings at each layer
  - Layer 0 embedding of node $u$ is its input features $X_u$

  **Basic neighborhood aggregation approach (i.e. $\square$):** Average information from neighbors and apply a neural network
Basic approach

Initial 0-th layer embeddings are equal to node features

\[ h^0_v = x_v \]

\[ h^k_v = \sigma \left( W_k \sum_{u \in N(v)} \frac{h^{k-1}_u}{|N(v)|} + B_k h^{k-1}_v \right), \quad \forall k \in \{1, \ldots, K\} \]

\[ z_v = h^K_v \]

Embedding after K layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

Previous layer embedding of \( v \)

Average of neighbor’s previous layer embeddings
We can feed these into any **loss function** and run **stochastic gradient descent** to train the **weight parameters**.

\[
\begin{align*}
\mathbf{h}_v^0 &= \mathbf{x}_v \\
\mathbf{h}_v^k &= \sigma \left( \mathbf{W}_k \sum_{u \in \mathcal{N}(v)} \frac{\mathbf{h}_u^{k-1}}{|\mathcal{N}(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1} \right), \quad \forall k \in \{1, \ldots, K\} \\
\mathbf{z}_v &= \mathbf{h}_v^K
\end{align*}
\]
Application: Prioritize drug combos

- Combinatorial explosion
  - >13 million possible combinations of 2 drugs
  - >20 billion possible combinations of 3 drugs
- Non-linear & non-additive interactions
  - Different effect than the additive effect of individual drugs
- Small subsets of patients
  - Side effects are interdependent
  - No info on drug combinations not yet used in patients

Modeling Polypharmacy Side Effects with Graph Convolutional Networks, Bioinformatics, 2018
Polypharmacy dataset

- Molecular, drug, and patient data for all drugs prescribed in US
  - 4,651,131 drug-drug edges: Patient data from adverse event system, tested for confounders [FDA]
  - 18,596 drug-protein edges
  - 719,402 protein-protein edges: Physical, metabolic enzyme-coupled, and signaling interactions
  - Drug and protein features: drugs’ chemical structure, proteins’ membership in pathways

- Gives multimodal network with over 5 million edges separated into 1,000 different edge types
Experimental setup

- **Two main stages:**
  1. Learn an embedding for every node in polypharmacy network
  2. Predict a score for every drug-drug, drug-protein, protein-protein pair in the test set based on the embeddings

Example: How likely will Simvastatin and Ciprofloxacin, when taken together, break down muscle tissue?
Application: Diagnose patients

- Phenotypes are observable characteristics resulting from interactions between genotypes, as well as environment
  - Physicians utilize standardized vocabulary of phenotypes to describe human diseases.
  - By modeling diseases as collections of associated phenotypes, we can diagnose patients based on their presenting symptoms

Medical History:
Has asthma?  
Other chronic issues?  
......

Symptoms:
Severe Cough  
Wheezing  
......
Application: Diagnose patients

- **Graph**: Consider a graph $G$ built from the standardized vocabulary of phenotypes:
  - Nodes: phenotypes; edges: relationships between phenotypes
  - Patient is a set of phenotypes, a subgraph $S$ in $G$

- **Learning Task**: Predict the disease (label) most consistent with the phenotype subgraph $S$
Problem formulation

- **Goal:** Learn subgraph embeddings such that the likelihood of preserving subgraph topology is maximized in the embedding space
  - $S_i$ and $S_j$ with similar subgraph topology should be embedded close together in the embedding space
Why are subgraphs challenging?

- Need to predict over structures of **varying size**:
  - How to represent subgraphs that **are not** \( k \)-hop neighborhoods?

- Rich connectivity patterns, both **internally** and **externally** through interactions with rest of \( G \):
  - How to inject this information into a GNN?

- Subgraphs can be:
  - **Localized** and reside in one region of the graph
  - **Distributed** across multiple local neighborhoods
Subgraph neural networks

- SubGNN specifies a neural message passing architecture that generates a $d_S$-dimensional subgraph representation $z_S$ for every subgraph $S$
- Two main stages:
  1. Propagate messages from anchor patches to subgraphs, and aggregate messages into a final subgraph embedding
  2. Route messages through 3 channels to capture subgraph topology

<table>
<thead>
<tr>
<th>SUB-GNN Channel</th>
<th>SUB-GNN Subchannel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position (P)</td>
<td>Distance between $S_i$’s components</td>
</tr>
<tr>
<td>Neighborhood (N)</td>
<td>Identity of $S_i$’s internal nodes</td>
</tr>
<tr>
<td>Structure (S)</td>
<td>Internal connectivity of $S_i$</td>
</tr>
<tr>
<td>Border (B)</td>
<td>Distance between $S_i$ and rest of $G$</td>
</tr>
<tr>
<td></td>
<td>Identity of $S_i$’s border nodes</td>
</tr>
<tr>
<td></td>
<td>Border connectivity of $S_i$</td>
</tr>
</tbody>
</table>
#1: Subgraph message passing

- Property \( x \)-specific messages \( m_x \) are propagated from anchor patches to subgraph components.
- Anchor patches are helper subgraphs randomly sampled from \( G \): patches \( A_P, A_N, \) and \( A_S \) for position, neighborhood and structure.

Property-specific messages \( m_x \) are propagated from anchor patches to subgraph components.

**Anchor patches** are helper subgraphs randomly sampled from \( G \): patches \( A_P, A_N, \) and \( A_S \) for position, neighborhood and structure.

The similarity function between subgraph component and an anchor patch is computed as:

\[
\text{MSG}_x^{A \rightarrow S} = \gamma_x(S^{(c)}, A_x) \cdot a_x
\]

The embedding of the anchor patch is:

\[
g_{x,c} = \text{AGG}_M(\{\text{MSG}_x^{A \rightarrow S^{(c)}} \mid A_x \in A_x\}),
\]

The property-specific representation of subgraph component at the previous layer that gets updated is:

\[
h_{x,c} \leftarrow \sigma(W_x \cdot [g_{x,c}; h_{x,c}]),
\]
#2: Property-aware routing

- SubGNN specifies three channels for position, neighborhood, and structure.

- Each channel $x$ has three key elements:
  - Similarity function $\gamma_x : (S^{(c)}, A_x) \rightarrow [0,1]$ to weigh messages exchanged between patches and subgraph components.
  - Anchor patch sampling function $\varphi_x : (G, S^{(c)}) \rightarrow A_x$ to sample patches from underlying graph.
  - Anchor patch encoder $\psi_x : A_x \rightarrow a_x$ to encode patches into embeddings $a_x$.
  - These functions can be learned or pre-defined.

- Channel outputs $z_x$ are concatenated to produce a final subgraph representation $z_S$. 

Subgraph Neural Networks, NeurIPS, 2020
Disease diagnosis datasets

- Each consists of a base graph and subgraphs with associated labels
  - HPO-METAB and HPO-NEURO are clinical diagnostic tasks with the Human Phenotypic Ontology (HPO) network as the base graph
  - They ask the following:
    - What is the subcategory of **metabolic disease** consistent with the phenotypes (i.e., phenotype subgraph)?
    - What is the subcategory of **neurological disease** consistent with the phenotypes (i.e., phenotype subgraph)?
Experimental setup

- Two main stages:
  1. Learn an embedding for every (patient) phenotype subgraph in HPO network
  2. Predict the subcategory of a metabolic or neurological disease for a patient based on their subgraph embedding

Check out the paper, GitHub, and datasets for details:

zitniklab.hms.harvard.edu/projects/SubGNN
This Tutorial

1. **Methods**: Network diffusion, shallow network embeddings, and graph neural networks

2. **Applications**: Fundamental biological discoveries and precision medicine

3. **Outlook**: Future directions and Q&A session

4. **Hands-on exercises**: Demos, implementation details, tools, and tips
Resources

- **Books & survey papers**
  - William Hamilton, *Graph Representation Learning* (morganclaypool.com/doi/abs/10.2200/S01045ED1V01Y20209AIM046)
  - Li et al., *Graph Representation Learning for Biomedicine* (arxiv.org/abs/2104.04883)

- **Keynotes**
  - Michael Bronstein, “Geometric Deep Learning: The Erlangen Programme of ML” (ICLR 2021 keynote) (youtube.com/watch?v=w6Pw4MOzMuo)

- **Software & packages**
  - PyTorch Geometric
  - NetworkX
  - Stanford Network Analysis Platform (SNAP)
Resources

- Conferences & summer schools
  - London Geometry and Machine Learning Summer School ([logml.ai](https://logml.ai))
  - Learning on Graphs Conference ([logconference.github.io](https://logconference.github.io))

- Tutorials & code bases
  - Zitnik Lab Graph ML Tutorials ([github.com/mims-harvard/graphml-tutorials](https://github.com/mims-harvard/graphml-tutorials))
  - Stanford University’s CS224 ([web.stanford.edu/class/cs224w](http://web.stanford.edu/class/cs224w))

- Datasets
  - Precision Medicine Oriented Knowledge Graph (PrimeKG) ([zitniklab.hms.harvard.edu/projects/PrimeKG](http://zitniklab.hms.harvard.edu/projects/PrimeKG))
  - Therapeutic Data Commons (TDC) ([tdcommons.ai](http://tdcommons.ai))
  - BioSNAP ([snap.stanford.edu/biodata/](http://snap.stanford.edu/biodata/))
  - Open Graph Benchmark (OGB) ([ogb.stanford.edu](http://ogb.stanford.edu))