Deep Learning for Network Biology

Marinka Zitnik and Jure Leskovec

Stanford University
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 2: Graph Neural Networks

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

\[ f(\cdot) = \text{Disease similarity network} \]

\[ = \text{2-dimensional node embeddings} \]

**Intuition:** Map nodes to d-dimensional embeddings such that similar nodes in the graph are embedded close together.
**Goal:** Map nodes so that *similarity in the embedding space* (e.g., dot product) approximates *similarity in the network*.
Embedding Nodes

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

Need to define!

Input network

encode nodes

\( ENC(u) \)

\( ENC(v) \)

d-dimensional embedding space

\( z_u \)

\( z_v \)
Two Key Components

- **Encoder:** Map a node to a low-dimensional vector:

  \[
  \text{ENC}(v) = z_v
  \]

  node in the input graph

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

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

  Similarity of \( u \) and \( v \) in the network
  dot product between node embeddings
So Far: Shallow Encoders

**Shallow encoders:**

- **One-layer of data transformation**
- A single hidden layer maps node $u$ to embedding $z_u$ via function $f$, e.g.,

$$z_u = f(z_v, v \in N_R(u))$$

Diagram:

- Node $u$
- Node $v$
- Embedding lookup
- Dot product $z_v^T z_u$
Shallow Encoders

Limitations of shallow encoding:

- \( O(|V|) \) parameters are needed:
  - No sharing of parameters between nodes
  - Every node has its own unique embedding

- Inherently “transductive”:
  - Cannot generate embeddings for nodes that are not seen during training

- Do not incorporate node features:
  - Many graphs have features that we can and should leverage
Deep Graph Encoders

Next: We will now discuss deep methods based on graph neural networks:

\[ \text{ENC}(v) = \text{multiple layers of non-linear transformation of graph structure} \]

Note: All these deep encoders can be combined with similarity functions from the previous section.
Deep Graph Encoders

Graph convolutions

Regularization, e.g., dropout

Graph convolutions

Activation function
Idea: Convolutional Networks

CNN on an image:

Goal is to generalize convolutions beyond simple lattices
Leverage node features/attributes (e.g., text, images)
Single CNN layer with 3x3 filter:

- Transform information at the neighbors and combine it:
  - Transform “messages” $h_i$ from neighbors: $W_i h_i$
  - Add them up: $\sum_i W_i h_i$
Real-World Graphs

But what if your graphs look like this?

or this:

- Examples:
  Biological networks, Medical networks, Social networks, Information networks, Knowledge graphs, Communication networks, Web graph, …
A Naïve Approach

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

Issues with this idea:
- $O(N)$ parameters
- Not applicable to graphs of different sizes
- Not invariant to node ordering
Outline of This Section

1. Basics of deep learning for graphs
2. Graph convolutional networks
3. Biomedical applications
Basics of Deep Learning for Graphs

Based on material from:


- Kipf et al., 2017. [Semisupervised Classification with Graph Convolutional Networks](https://arxiv.org/abs/1609.02907). *ICLR*.
Assume we have a graph $G$:

- $V$ is the vertex set
- $A$ is the adjacency matrix (assume binary)
- $X \in \mathbb{R}^{m \times |V|}$ is a matrix of node features
  - Biologically meaningful node features:
    - E.g., immunological signatures, gene expression profiles, gene functional information
  - No features:
    - Indicator vectors (one-hot encoding of a node)
Examples

Protein-protein interaction networks in different tissues, e.g., blood, substantia nigra

**Node feature:** Associations of proteins with angiogenesis

**Node feature:** Associations of proteins with midbrain development
Graph Convolutional Networks

Problem: For a given subgraph how to come with canonical node ordering

Our Approach

Idea: Node’s neighborhood defines a computation graph

Determine node computation graph
Propagate and transform information

Learn how to propagate information across the graph to compute node features

Semi-Supervised Classification with Graph Convolutional Networks. T. N. Kipf, M. Welling, ICLR 2017
Idea: Aggregate Neighbors

Key idea: Generate node embeddings based on local network neighborhoods

INPUT GRAPH

TARGET NODE

Deep Learning for Network Biology -- snap.stanford.edu/deepnetbio-ismb -- ISMB 2018
Idea: Aggregate Neighbors

Intuition: Nodes aggregate information from their neighbors using neural networks

TARGET NODE

INPUT GRAPH

Neural networks
Idea: Aggregate Neighbors

**Intuition:** Network neighborhood defines a computation graph

Every node defines a computation graph based on its neighborhood!
Deep Model: Many Layers

- Model can be of arbitrary depth:
  - Nodes have embeddings at each layer
  - Layer-0 embedding of node $u$ is its input feature, i.e. $x_u$. 
Neighborhood aggregation: Key distinctions are in how different approaches aggregate information across the layers.
Neighborhood Aggregation

- **Basic approach**: Average information from neighbors and apply a neural network

1) average messages from neighbors

2) apply neural network
Basic approach: Average neighbor messages and apply a neural network

Initial 0-th layer embeddings are equal to node features

Embedding after K layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

Previous layer embedding of \( v \)

Average of neighbor’s previous layer embeddings

\[
\begin{align*}
\mathbf{h}_v^0 &= \mathbf{x}_v \\
\mathbf{h}_v^k &= \sigma \left( W_k \sum_{u \in N(v)} \frac{\mathbf{h}_{u}^{k-1}}{|N(v)|} + B_k h_v^{k-1} \right), \quad \forall k \in \{1, \ldots, K\}
\end{align*}
\]
Training the Model

How do we train the model to generate embeddings?

Need to define a loss function on the embeddings!

Need to define a loss function on the embeddings!
We can feed these embeddings into any loss function and run stochastic gradient descent to train the weight parameters.
Unsupervised Training

- Train in an **unsupervised manner**:
  - Use only the graph structure
  - "Similar" nodes have similar embeddings
- Unsupervised loss function can be anything from the last section, e.g., a loss based on
  - Random walks (node2vec, DeepWalk, struc2vec)
  - Graph factorization
  - Node proximity in the graph
Unsupervised: Example

**Supervised Training**

**Directly train** the model for a supervised task (e.g., node classification)

- Safe or toxic drug?
- Safe or toxic drug?
- E.g., a drug-drug interaction network

---

Deep Learning for Network Biology -- snap.stanford.edu/deepnetbio-ismb -- ISMB 2018
Graph neural network applied to gene-gene interaction graph to predict gene expression level

Single gene inference task by adding nodes based on their distance from the node we want to predict

Training the Model

Directly train the model for a supervised task (e.g., node classification)

\[ \mathcal{L} = \sum_{v \in V} y_v \log(\sigma(z_v^\top \theta)) + (1 - y_v) \log(1 - \sigma(z_v^\top \theta)) \]

Encoder output: node embedding

Safe or toxic drug?

Classification weights

Node class label
Model Design: Overview

1) Define a neighborhood aggregation function

2) Define a loss function on the embeddings
3) Train on a set of nodes, i.e., a batch of compute graphs
4) Generate embeddings for nodes
Summary So Far

- **Recap:** Generate node embeddings by aggregating neighborhood information
  - We saw a **basic variant of this idea**
  - Key distinctions are in how different approaches aggregate information across the layers

- **Next:** Describe state-of-the-art graph neural network
Outline of This Section

1. Basics of deep learning for graphs
2. Graph convolutional networks
3. Biomedical applications
Graph Convolutional Networks

Based on material from:
• Hamilton et al., 2017. Inductive Representation Learning on Large Graphs. NIPS.
So far we have aggregated the neighbor messages by taking their (weighted) average. Can we do better?
GraphSAGE: Idea

Any differentiable function that maps set of vectors in $N(u)$ to a single vector

$$h^k_v = \sigma \left( \left[ A_k \cdot \text{AGG}(\{h^{k-1}_u, \forall u \in N(v)\}), B_k h^{k-1}_v \right]\right)$$
GraphSAGE Aggregation

- Simple neighborhood aggregation:

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

- GraphSAGE:

\[
    h^k_v = \sigma \left( [W_k \cdot \text{AGG} \left( \{h^{k-1}_u, \forall u \in N(v)\} \right)] , B_k h^{k-1}_v \right)
\]

Concatenate self embedding and neighbor embedding

generalized aggregation
Variants of Aggregation

**Mean:** Take a weighted average of neighbors

\[
\text{AGG} = \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|}
\]

**Pool:** Transform neighbor vectors and apply symmetric vector function

\[
\text{AGG} = \gamma(\{Qh_u^{k-1}, \forall u \in N(v)\})
\]

**LSTM:** Apply LSTM to reshuffled of neighbors

\[
\text{AGG} = \text{LSTM} \left( [h_u^{k-1}, \forall u \in \pi(N(v))] \right)
\]
Key idea: Generate node embeddings based on local neighborhoods

- Nodes aggregate “messages” from their neighbors using neural networks
More on Graph Neural Nets

Attention-based neighborhood aggregation:
- Graph attention networks (Hoshen, 2017; Velickovic et al., 2018; Liu et al., 2018)

Embedding edges and entire graphs:
- Graph neural nets with edge embeddings (Battaglia et al., 2016; Gilmer et. al., 2017)
- Embedding entire graphs (Duvenaud et al., 2015; Dai et al., 2016; Li et al., 2018)

Spectral approaches to graph neural networks:
- Spectral graph CNN & ChebNet (Bruna et al., 2015; Defferrard et al., 2016)

Hyperbolic geometry and hierarchical embeddings:
- Hierarchical relations (Nickel et al., 2017; Nickel et al., 2018)
Outline of This Section

1. Basics of deep learning for graphs
2. Graph convolutional networks
3. Biomedical applications
Application: Tissue-specific Protein Function Prediction

Material based on:
- Zitnik and Leskovec. 2017. Predicting Multicellular Function through Multilayer Tissue Networks. ISMB.
- Hamilton et al., 2017. Inductive Representation Learning on Large Graphs. NIPS.
Why Protein Functions?

Knowledge of protein functions in different tissues is essential for:

- Understanding human biology
- Interpreting genetic variation
- Developing disease treatments

[Greene et al. 2015, Yeger & Sharan 2015, GTEx and others]
Why Predicting Protein Functions?

Biotechnological limits & rapid growth of sequence data: most proteins can only be annotated computationally
Protein Function Prediction

This is a multi-label node classification task

Cell cycle

CLB4  CDC16  UNK1

RPT1  RPN3  UNK2

Cell proliferation

CLB4  CDC16  UNK1

RPT1  RPN3  UNK2

Machine Learning

CDC3  UNK2

Deep Learning for Network Biology -- snap.stanford.edu/deepnetbio-ismb -- ISMB 2018
What Does My Protein Do?

**Goal:** Given a protein and a tissue, predict the protein’s functions in that tissue

Proteins × Functions × Tissues → [0,1]

- **Substantia nigra tissue**

  - Midbrain development
  - WNT1

- **Blood tissue**

  - Angiogenesis
  - RPT6

\[ WNT1 \times (\text{Midbrain development, Substantia nigra}) \rightarrow 0.9 \]
\[ RPT6 \times (\text{Angiogenesis, Blood}) \rightarrow 0.05 \]
Existing Research

- Guilty by association: protein’s function is determined based on who it interacts with
  - No tissue-specificity
- Protein functions are assumed constant across organs and tissues:
  - Functions in heart are the same as in skin

Lack of methods for predicting protein functions in different biological contexts
Challenges

- Tissues are related to each other:
  - Proteins in biologically similar tissues have similar functions
  - Proteins are missing in some tissues
- Little is known about tissue-specific protein functions:
  - Many tissues have no annotations
Approach

1. Represent every tissue with a separate protein-protein interaction graph:
   - Protein function prediction is a multi-label node classification task
   - Each protein can have 0, 1, or more functions (labels) in each tissue

2. Learn protein embeddings:
   - Use PPI graphs and labels to train GraphSAGE:
     - Learn how to embed proteins in each tissue:
       - Aggregate neighborhood information
       - Share parameters in the encoder
     - Use inductive learning!
The same aggregation parameters are shared for all nodes:

- Can generalize to unseen nodes
- Can make predictions on entirely unseen graphs (tissues)!
Inductive Learning of Tissues

1. Train on a protein-protein interaction graph from one tissue
2. Generate embeddings and make predictions for newly collected data about a different tissue

Inductive node embedding generalize to entirely unseen graphs

Midbrain development

Train on forebrain tissue

Generalize to blood tissue

Angiogenesis

WNT1

RPT6
Data and Setup

- **Data:**
  - Protein-protein interaction (PPI) graphs, with each graph corresponding to a different human tissue
  - Use positional gene sets, motif gene sets, and immunological signatures from MSigDB as node features
    - Feature data is very sparse (42% of nodes have no features)
    - This makes leveraging neighborhood information critical
  - Use Gene Ontology annotations as labels

- **Setup:**
  - Multi-label node classification:
    - Each protein can have 0, 1, or more functions (labels) in each tissue
  - Train GraphSAGE on 20 tissue-specific PPI graphs
  - Generate new embeddings “on the fly”
  - Make prediction on entirely unseen graphs (i.e., new tissues)
Annotating New Tissues

- **Transfer** protein functions to an **unannotated tissue**
- **Task:** Predict functions in target tissue without access to any annotation/label in that tissue

<table>
<thead>
<tr>
<th>Name</th>
<th>Unsup. F1</th>
<th>Sup. F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>0.396</td>
<td>0.396</td>
</tr>
<tr>
<td>Raw features</td>
<td>0.422</td>
<td>0.422</td>
</tr>
<tr>
<td>DeepWalk</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>DeepWalk + features</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>GraphSAGE-GCN</td>
<td>0.465</td>
<td>0.500</td>
</tr>
<tr>
<td>GraphSAGE-mean</td>
<td>0.486</td>
<td>0.598</td>
</tr>
<tr>
<td>GraphSAGE-LSTM</td>
<td>0.482</td>
<td><strong>0.612</strong></td>
</tr>
<tr>
<td>GraphSAGE-pool</td>
<td><strong>0.502</strong></td>
<td>0.600</td>
</tr>
</tbody>
</table>

- GraphSAGE significantly outperforms the baseline approaches
- LSTM- and pooling-based aggregators outperform mean- and GCN-based aggregators

Unsup. – unsupervised; Sup. – fully supervised

GraphSAGE F1 – scores are in [0,1], higher is better
Outline of This Section

1. Basics of deep learning for graphs
2. Graph convolutional networks
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