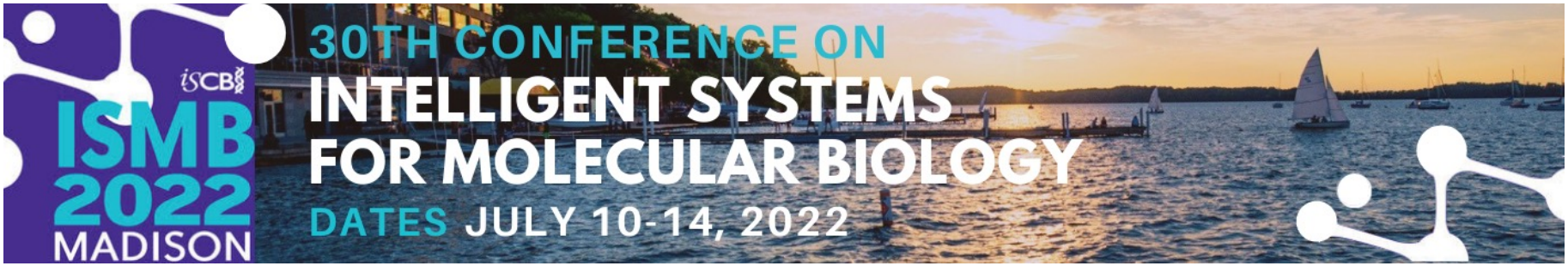
Towards Precision Medicine with Graph Representation Learning

Michelle M. Li & Marinka Zitnik

Department of Biomedical Informatics
Broad Institute of Harvard and MIT
Harvard Data Science

zitniklab.hms.harvard.edu/biomedgraphml



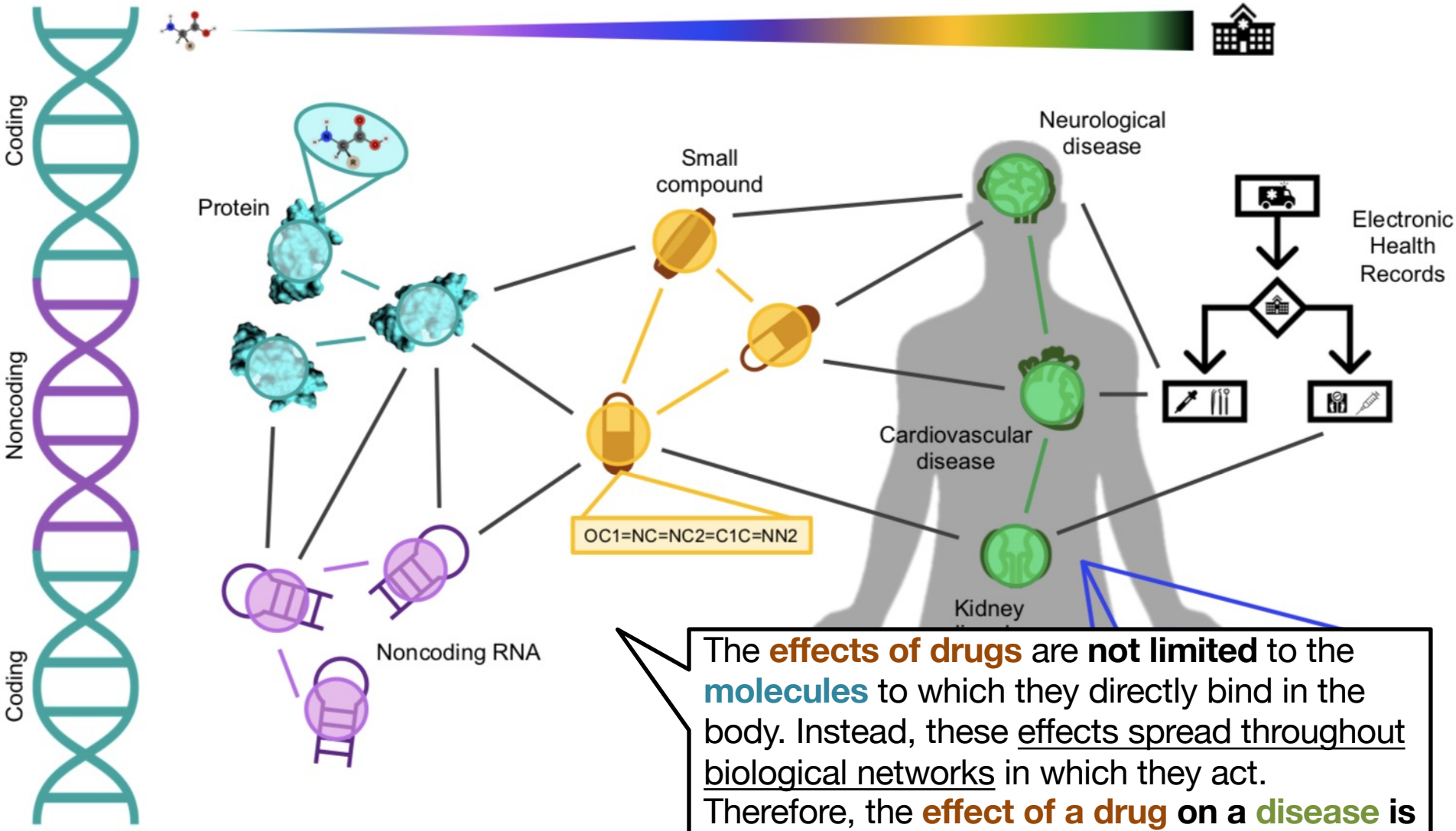
Tutorial VT4

July 7, 2022 at 9am – 1pm CDT

All tutorial materials are available at

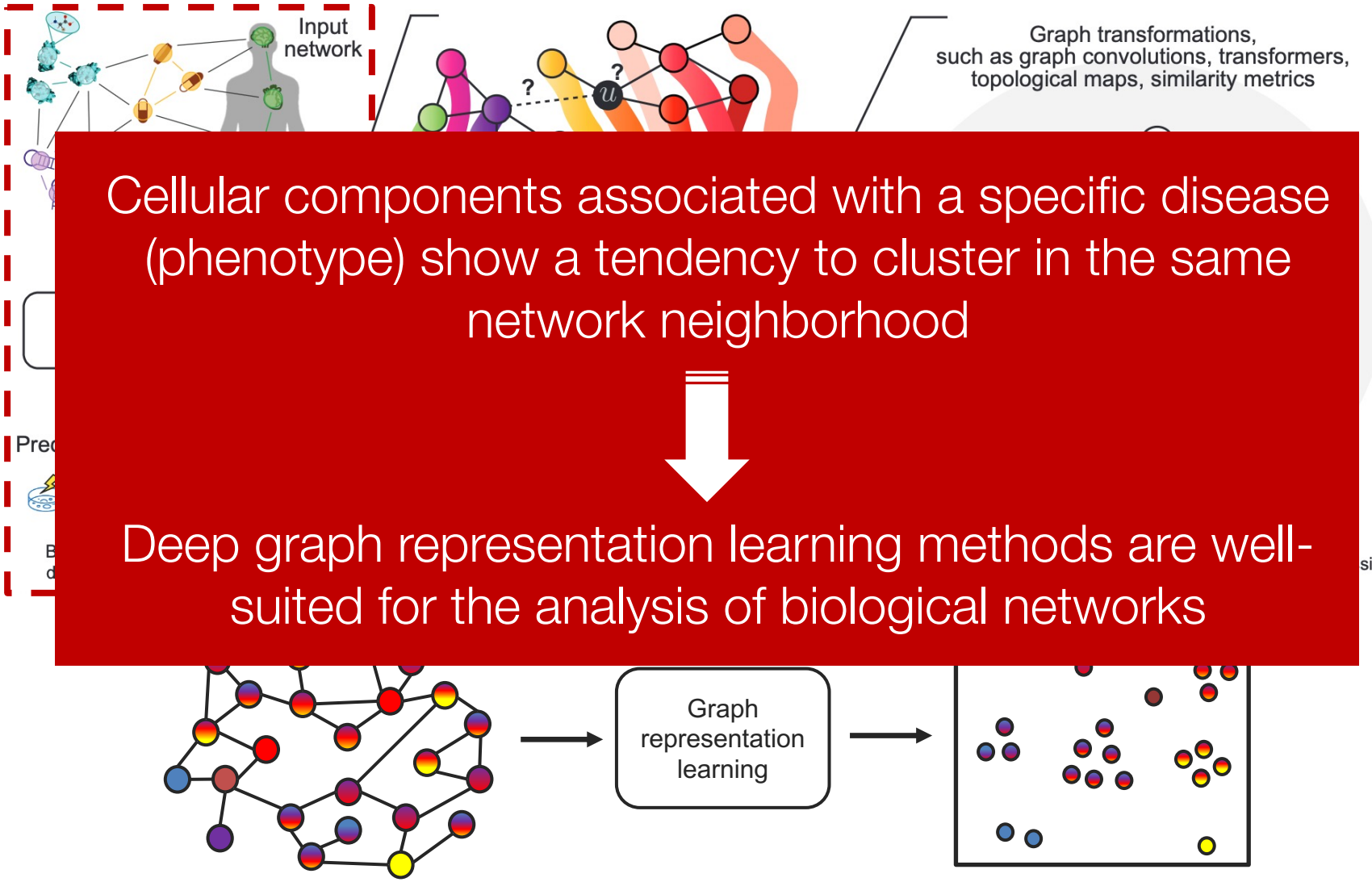
zitniklab.hms.harvard.edu/biomedgraphml

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



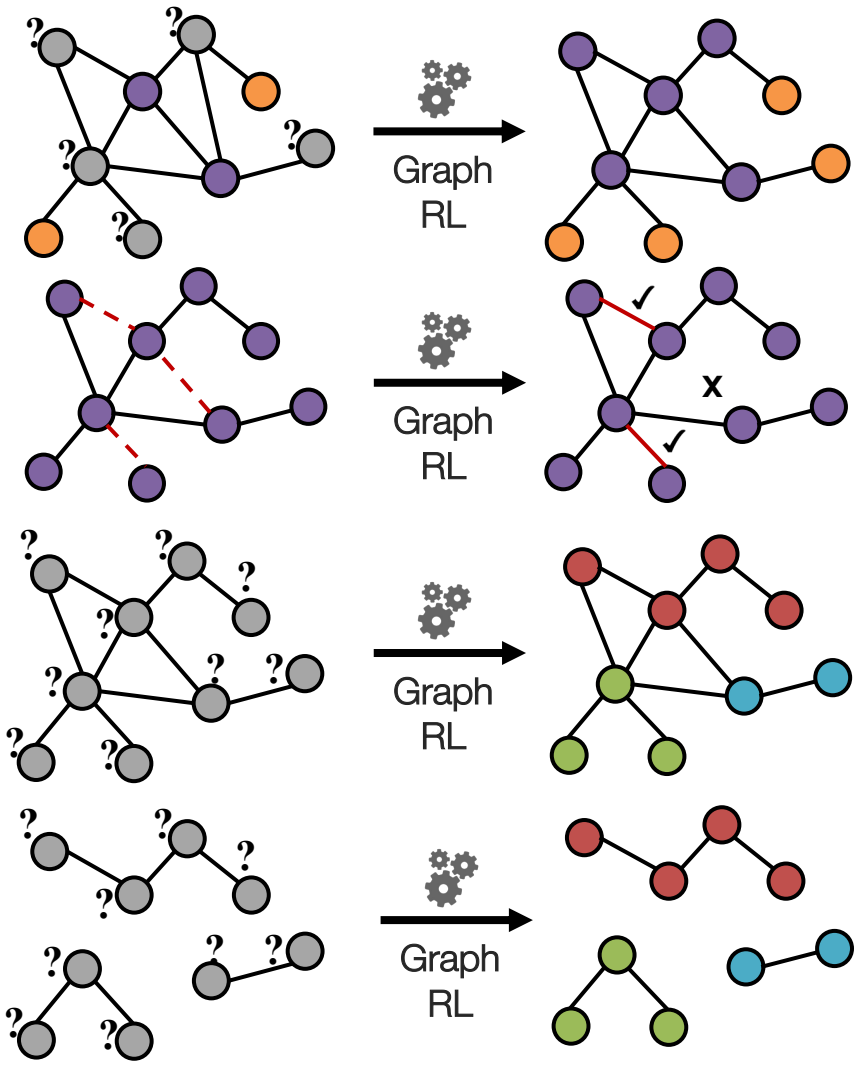
Graph Representation Learning for Biomedicine, *Nature Biomedical Engineering* (in press), 2022, arXiv:2104.04883

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

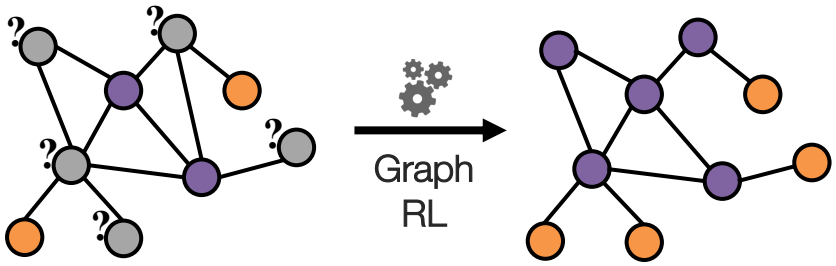
Graph representation learning tasks

- 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 tasks

- Node-level
  - Characteristics of a given node



Node classification: Example

Classifying the function of proteins in the interactome!

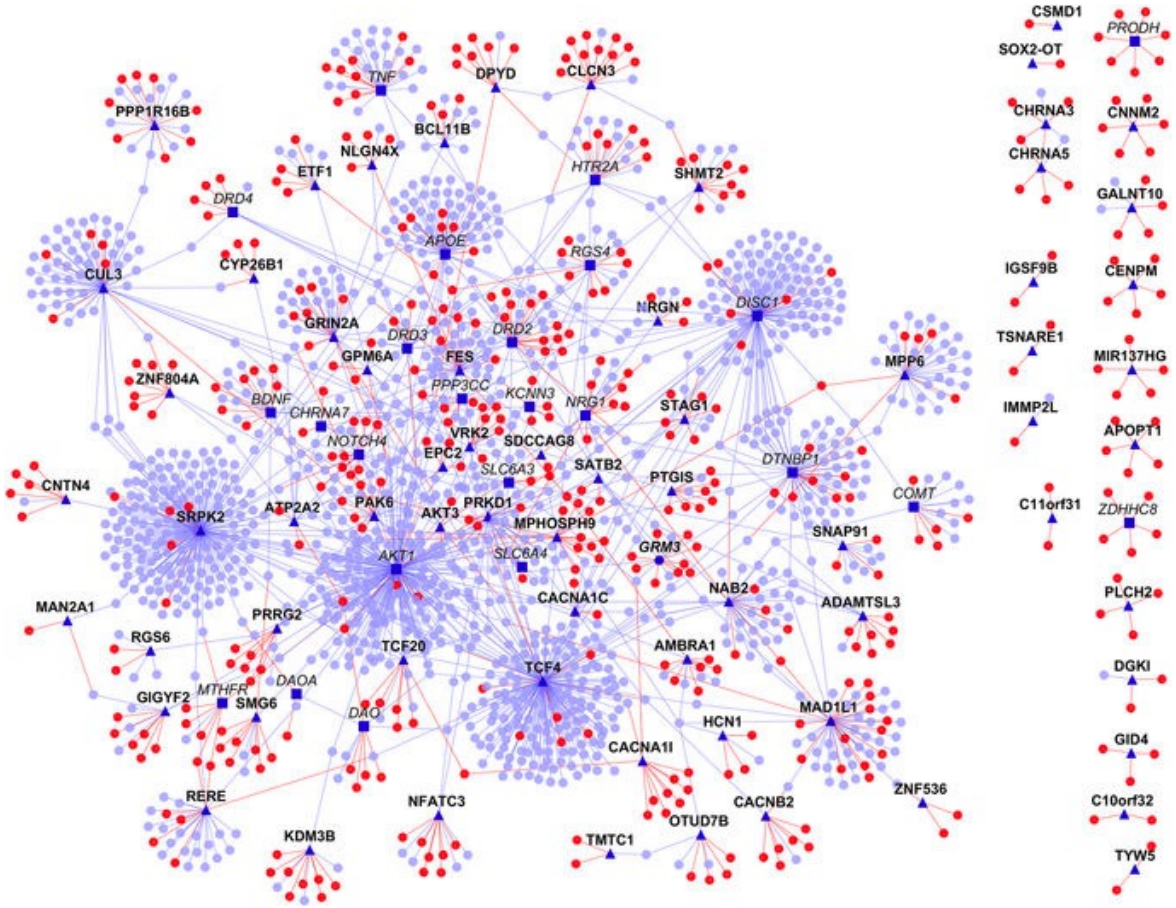
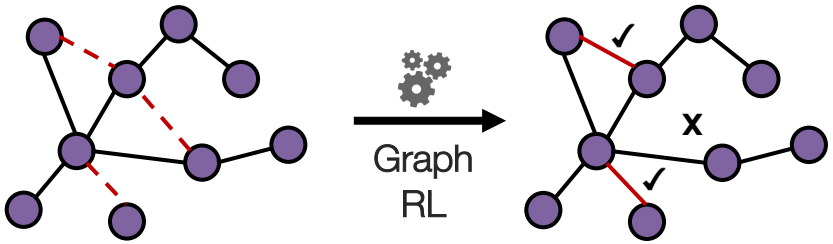


Image from Ganapathiraju et al. 2016. Schizophrenia interactome with 504 novel protein–protein interactions. *Nature*.

Graph representation learning tasks

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



Link prediction: Example

Predicting which diseases a new molecule might treat!

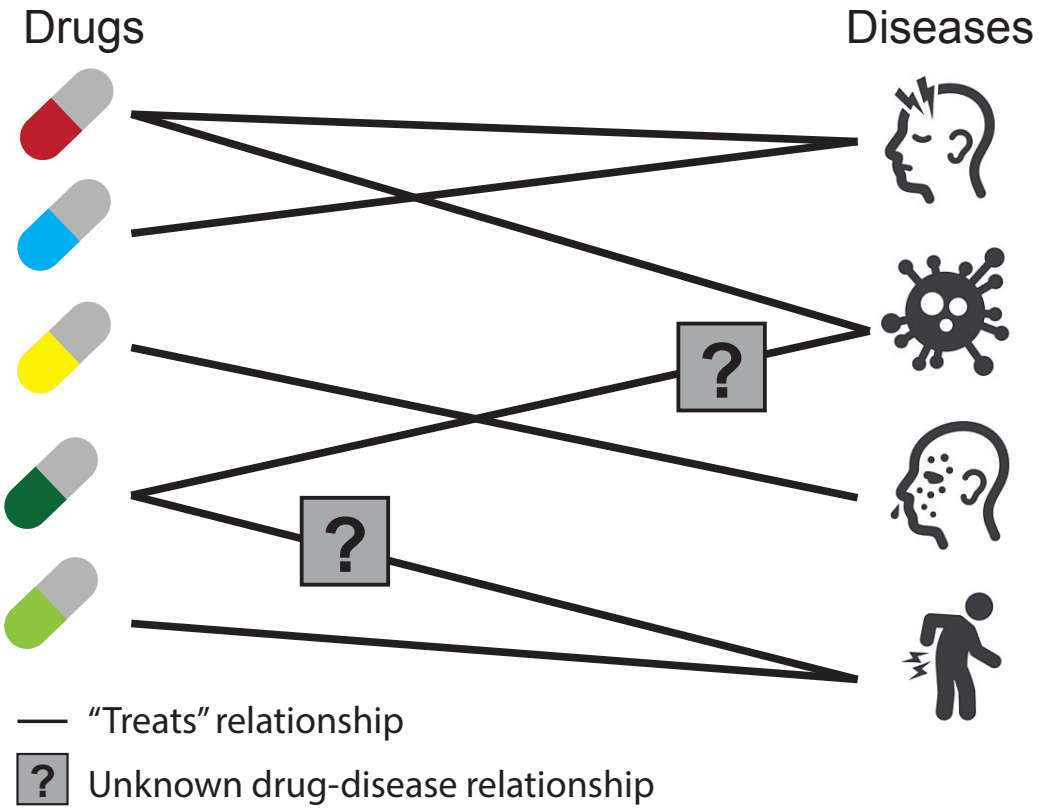
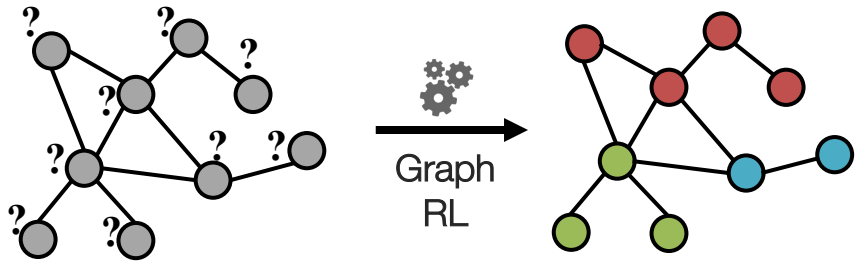


Image from: Zitnik et al. 2020. Network-based discovery of drug indications.

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!

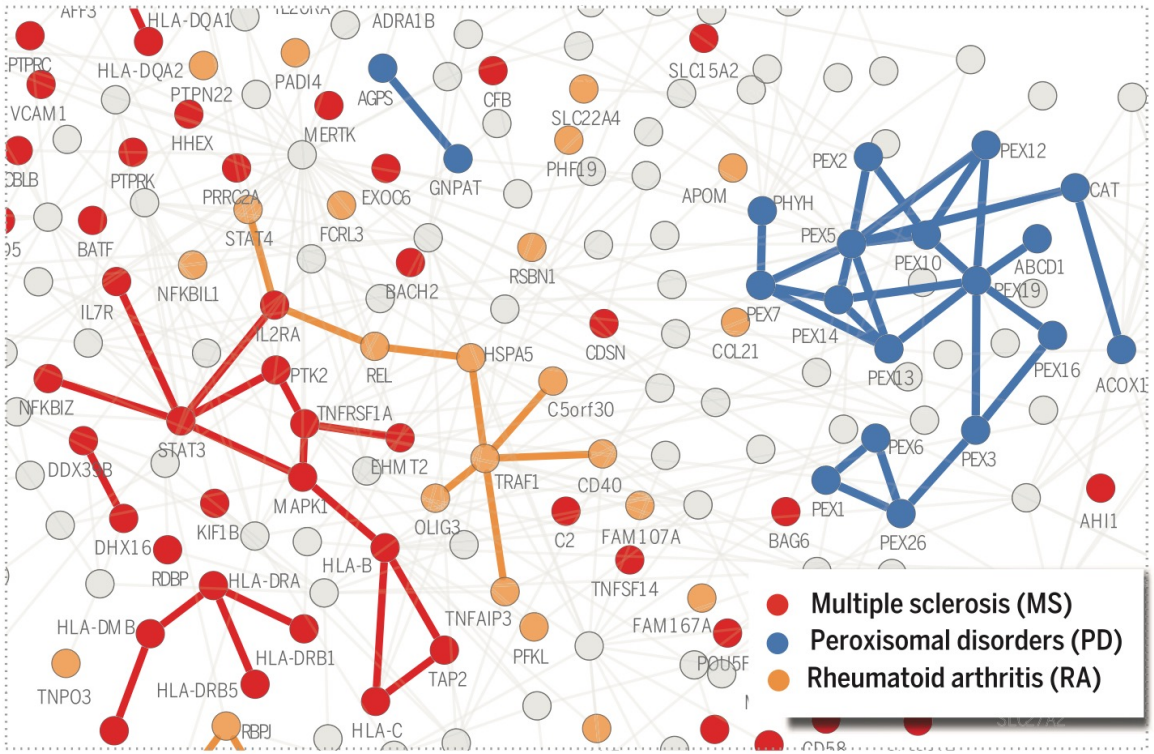
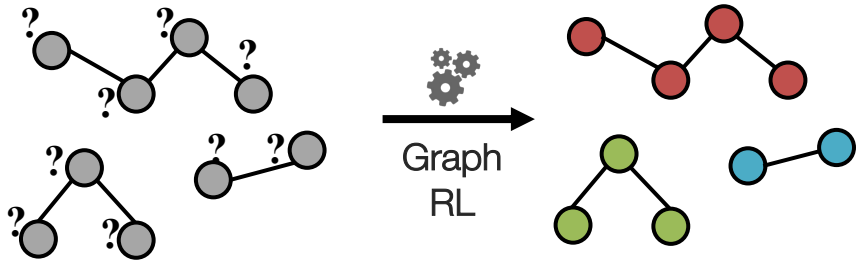


Image from: Menche et al. 2015. Uncovering disease-disease relationships through the incomplete interactome. *Science*.

Graph representation learning tasks

- Graph-level
  - Similarity of graph to other graphs



Graph classification: Example

Designing new small molecule compounds to treat a disease!

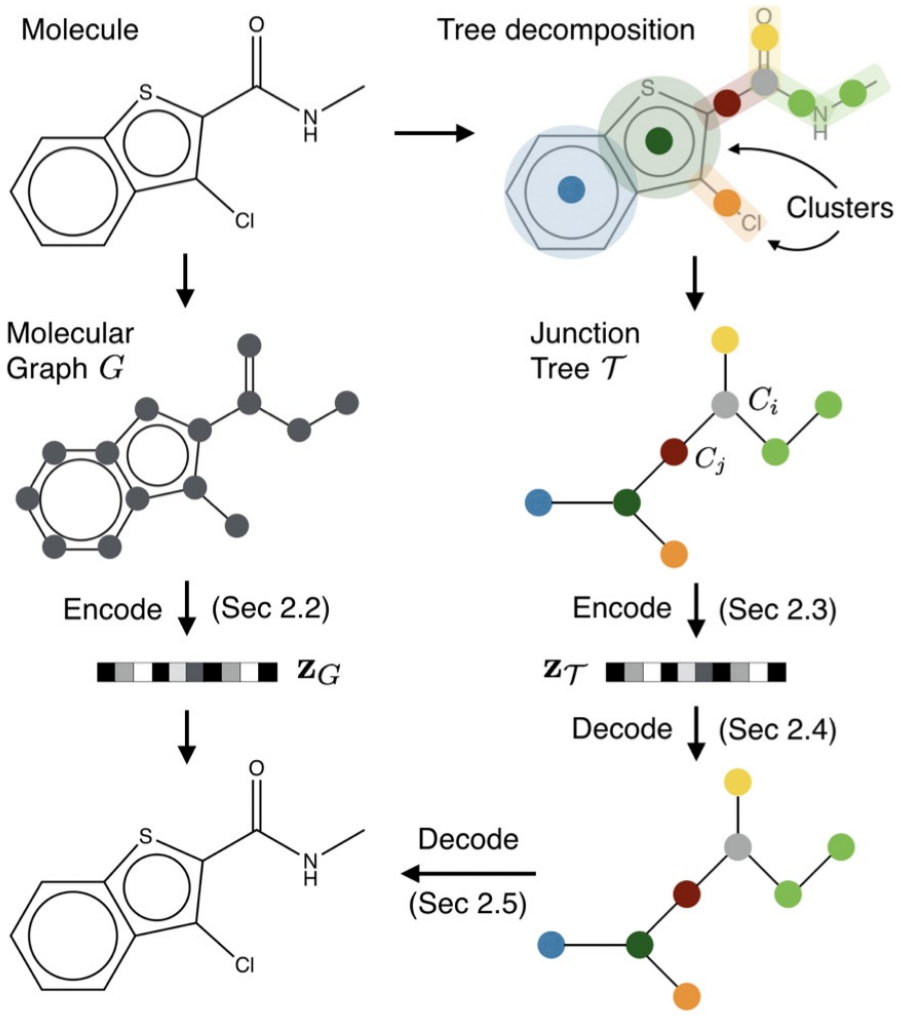
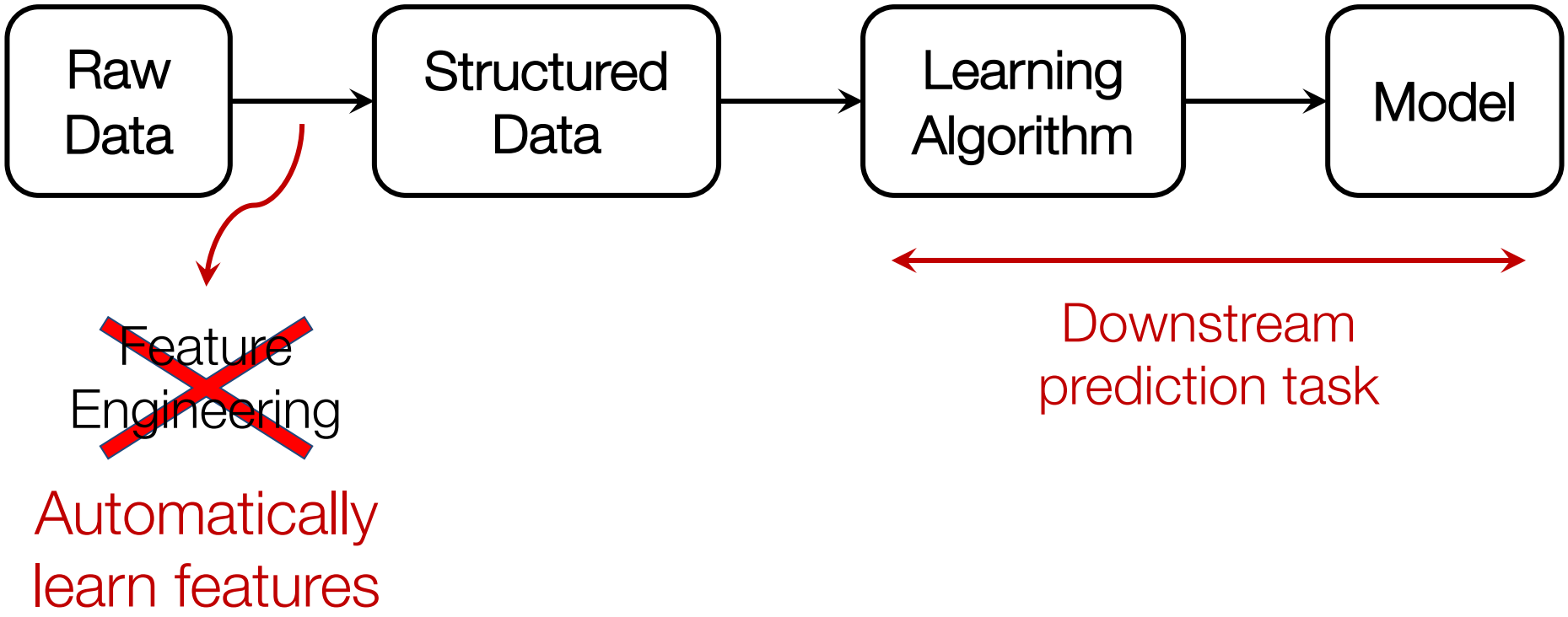


Image from: Jin et al. 2018. Junction Tree Variational Autoencoder for Molecular Graph Generation. *ICML.*

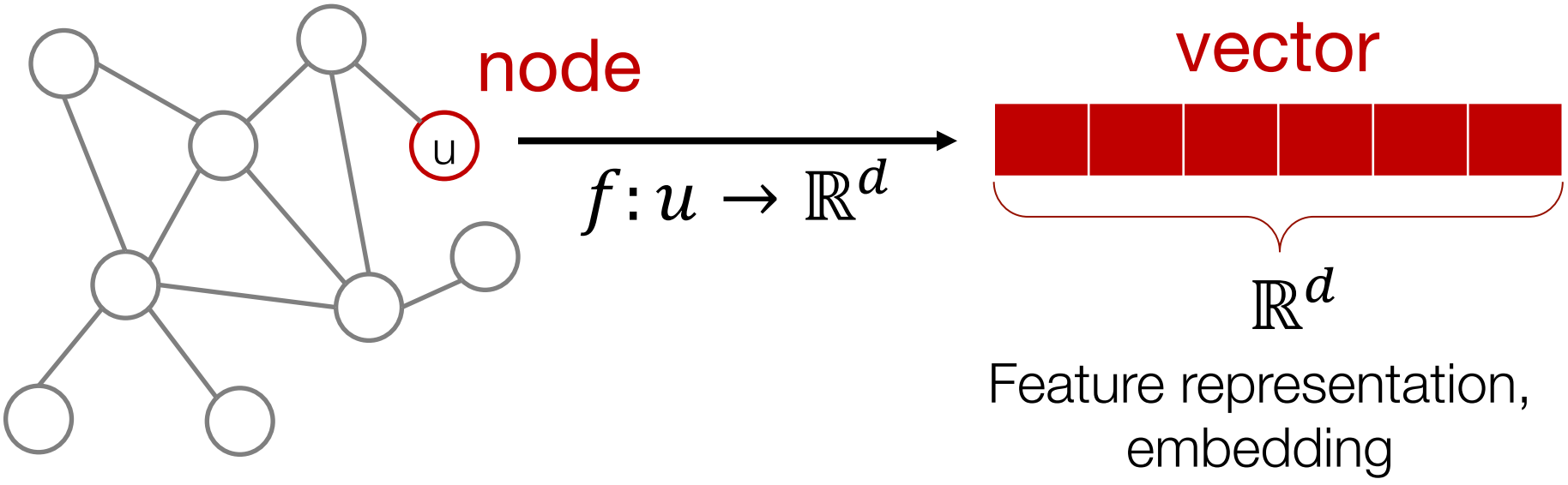
Predictive modeling lifecycle

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



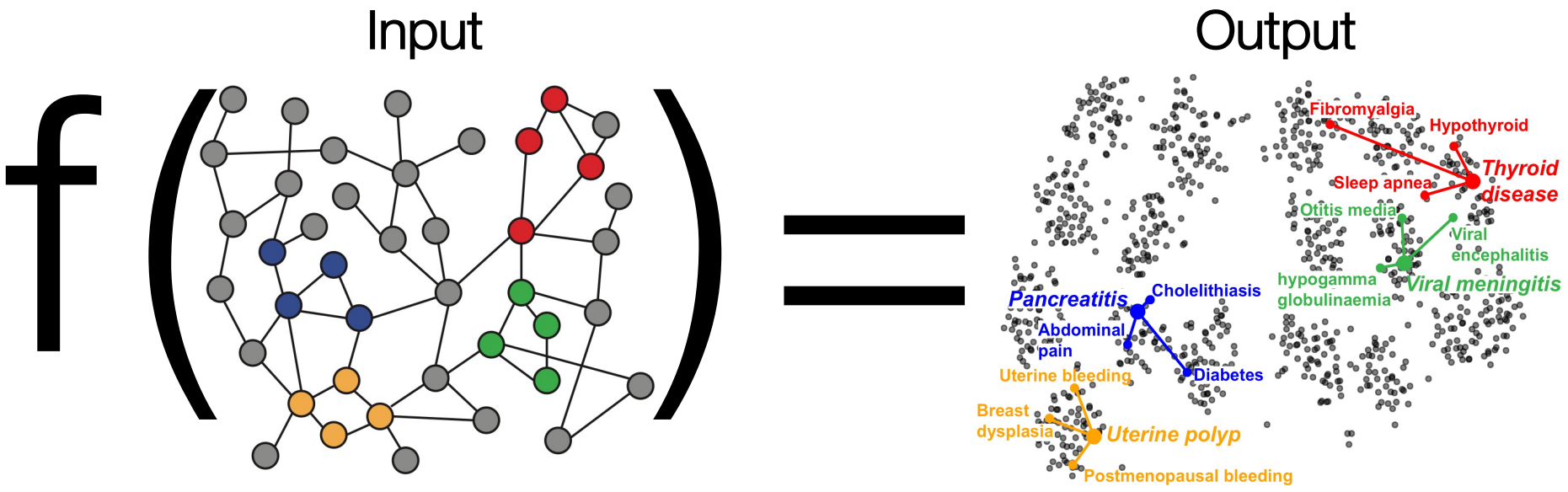
Feature learning in graphs

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



Embedding nodes

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

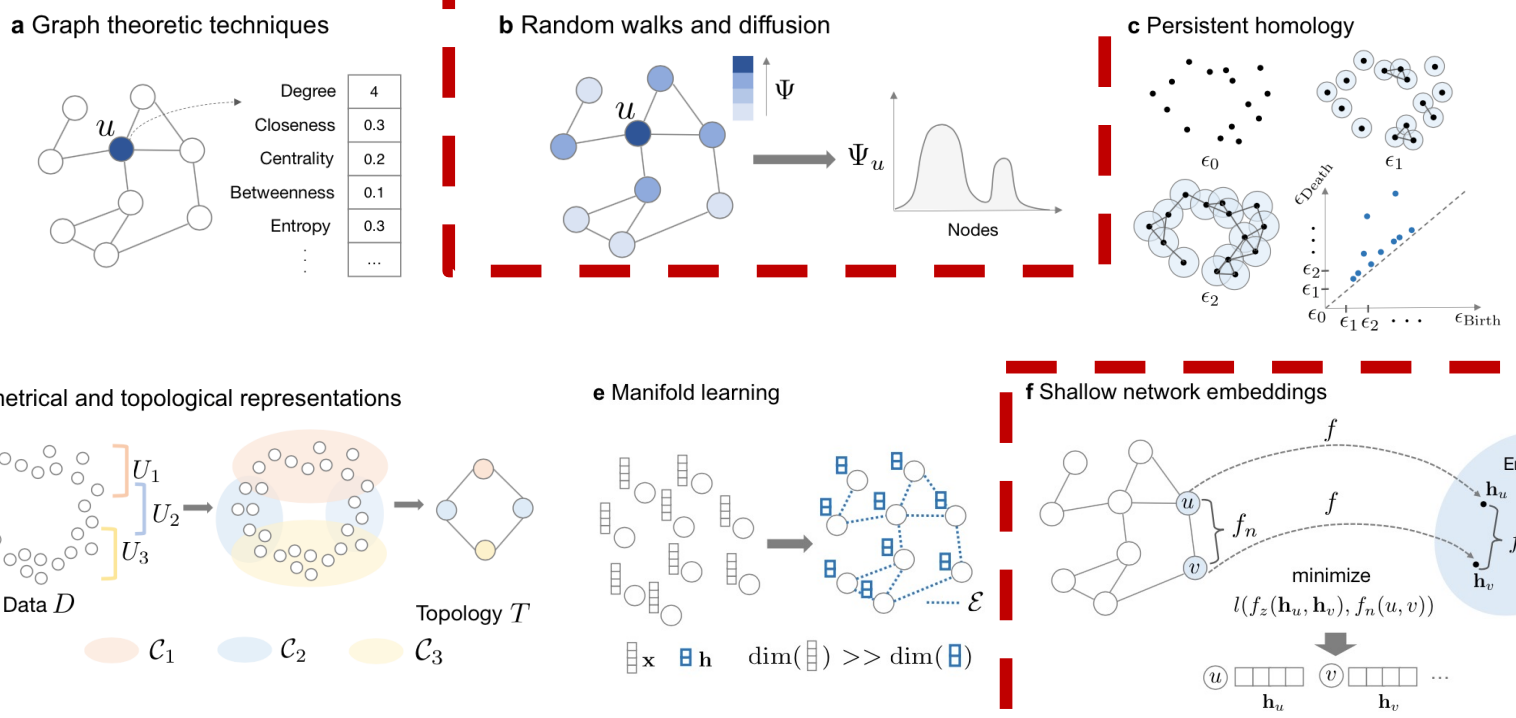


Disease similarity network

2D node embeddings

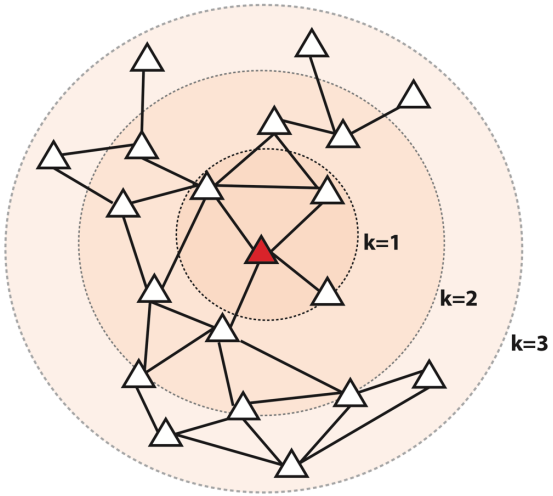
How to learn mapping function f?

Predominant graph learning paradigms



Random walks and diffusion

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

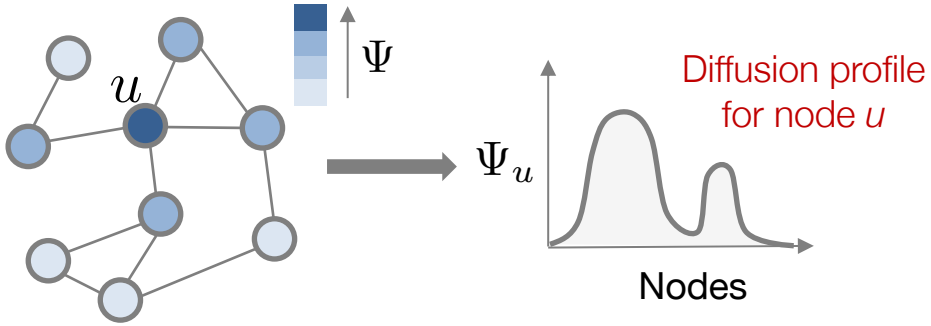


Red: Target node
k = 1: 1-hop neighbors
k = 2: 2-hop neighbors
k = 3: 3-hop neighbors

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

Random walks and diffusion

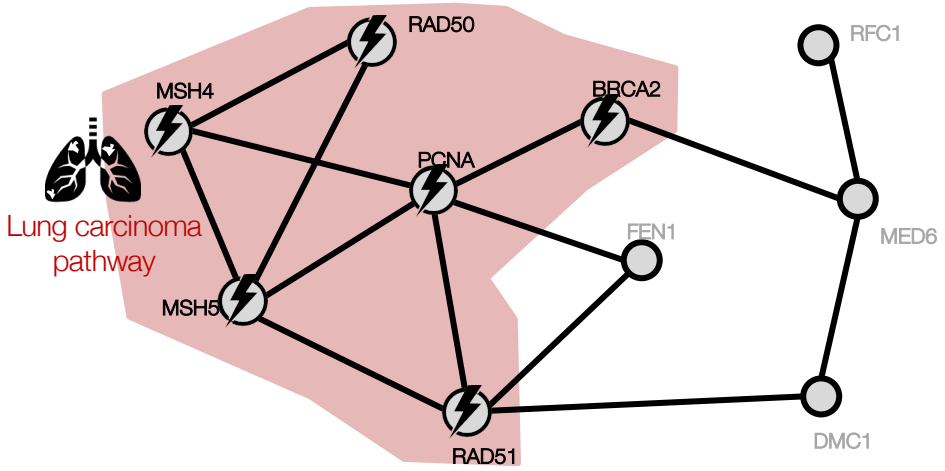
- Method: Diffusion state distance (DSD)
  - Simulate random walks from source node *u*
  - Count the number of random walks of length *k* that start at *u* and visit a destination node *v*
- Node representation: Each node *u* has vector Ψ_u that represents its influence on its *k*-hop neighborhood

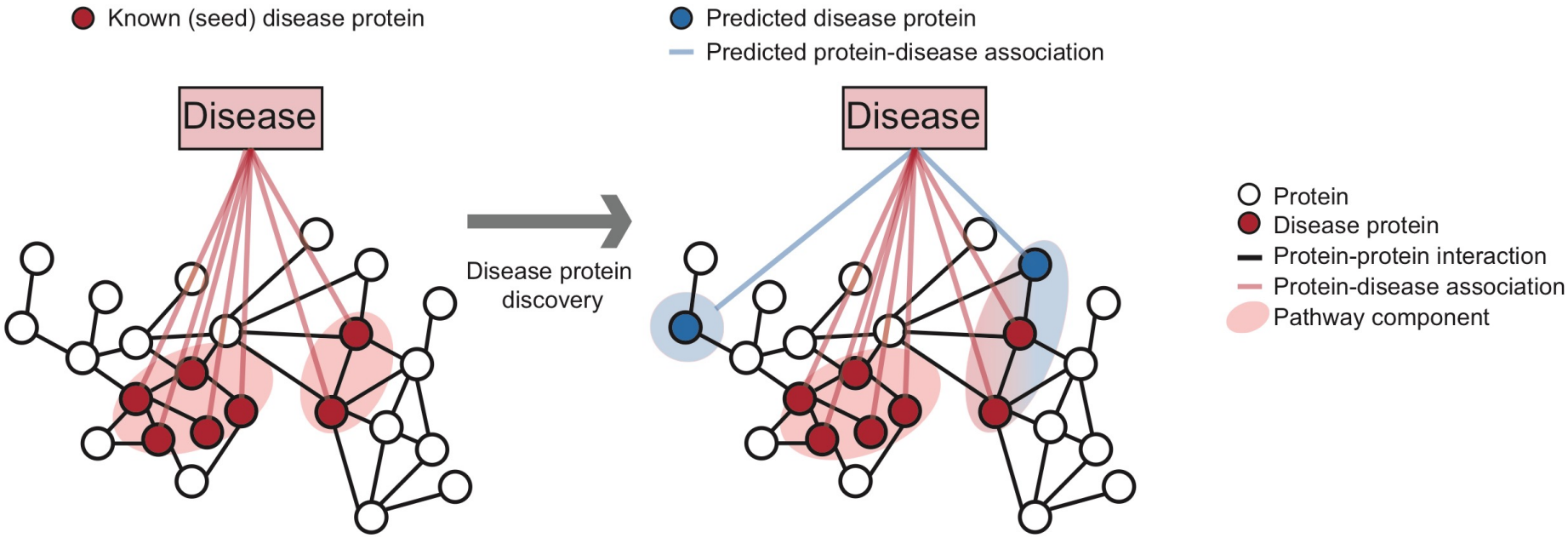


- Comparison: Calculate whether nodes *u* and *v* have similar local connectivity by $DSD(u, v) = \| \Psi_u - \Psi_v \|_1$

Application: Identify disease pathways

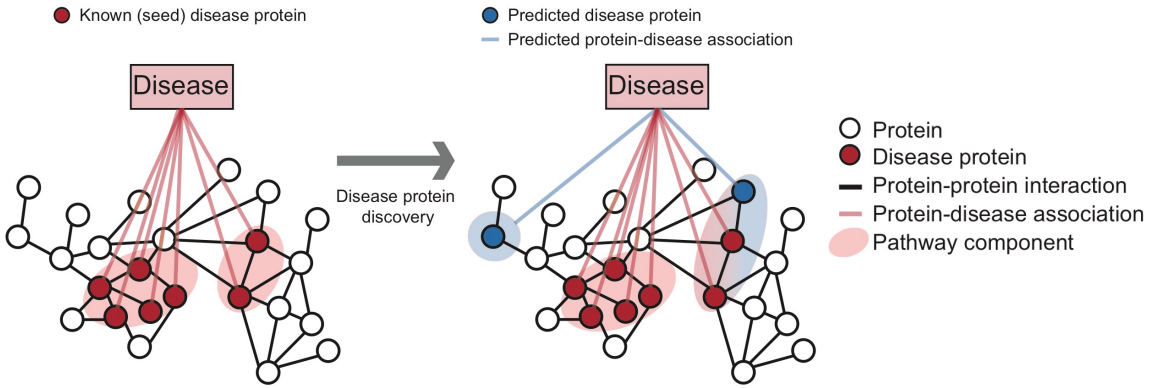
- Pathway: Subnetwork of interacting proteins associated with a disease





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

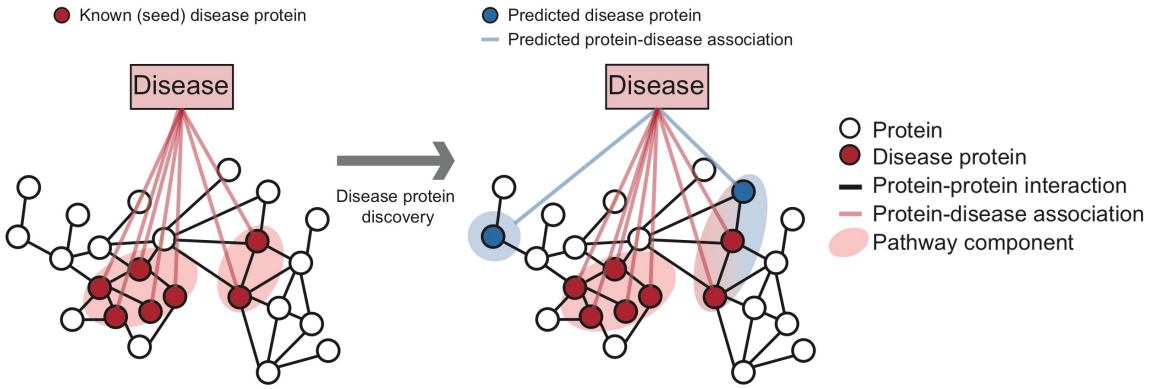
Disease pathway dataset



- 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*

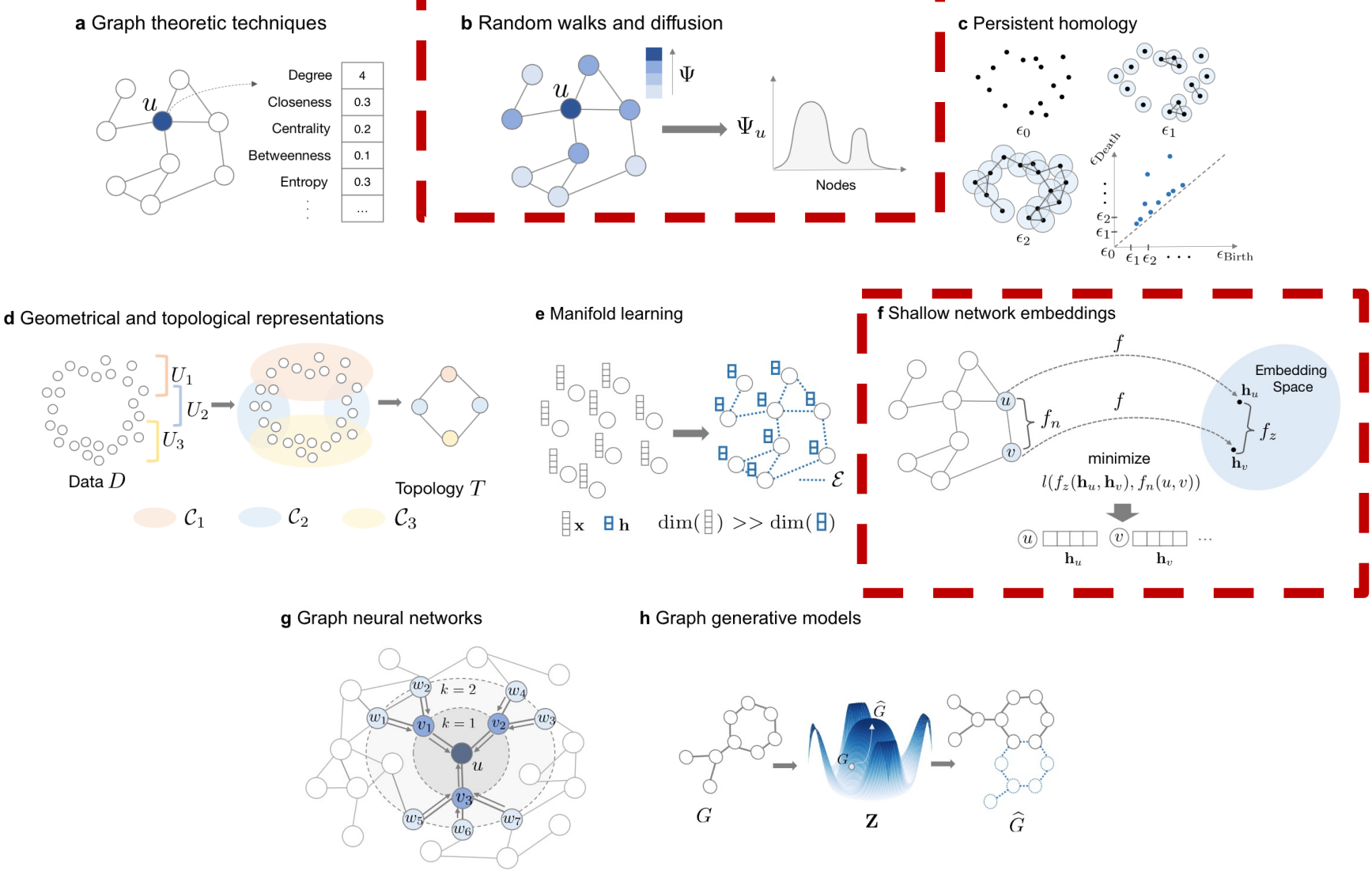
Experimental setup



- 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)

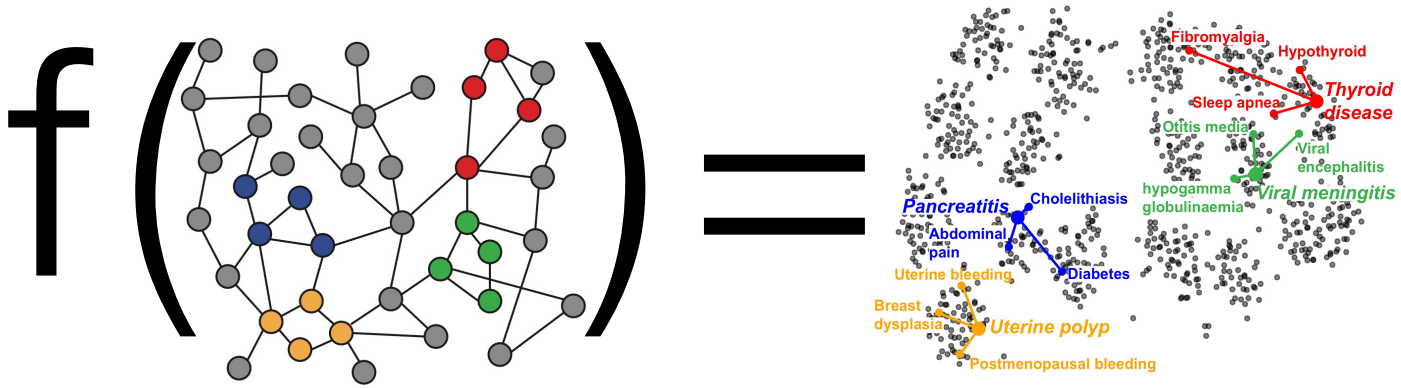
Large-scale analysis of disease pathways in the human interactome, *Pacific Symposium on Biocomputing, 2018*

Predominant graph learning paradigms



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 $\mathbf{G}$:
  - $\mathbf{V}$ is the vertex set
  - $\mathbf{A}$ is the adjacency matrix (assume binary)
  - No node features or extra information is used!

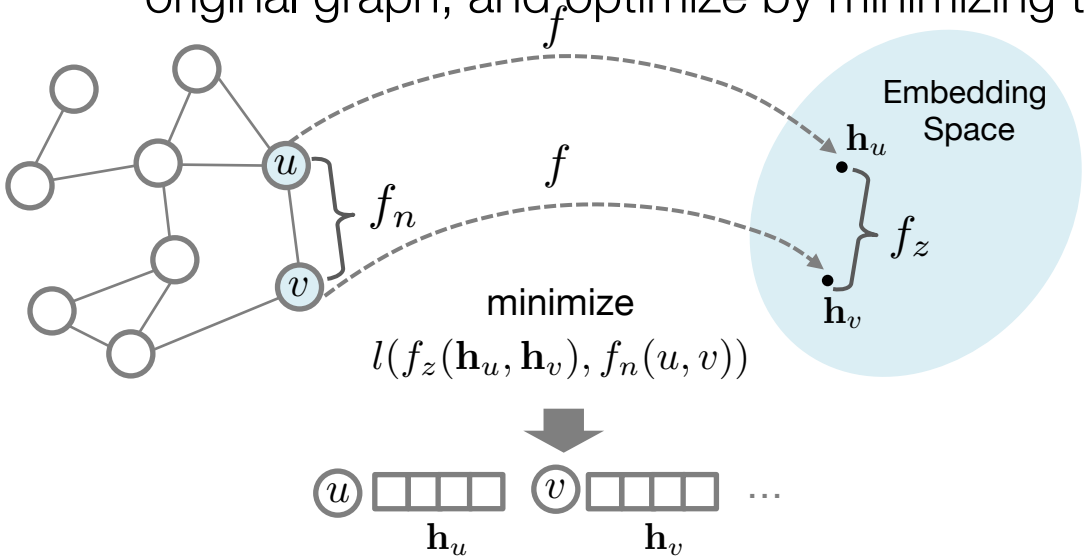


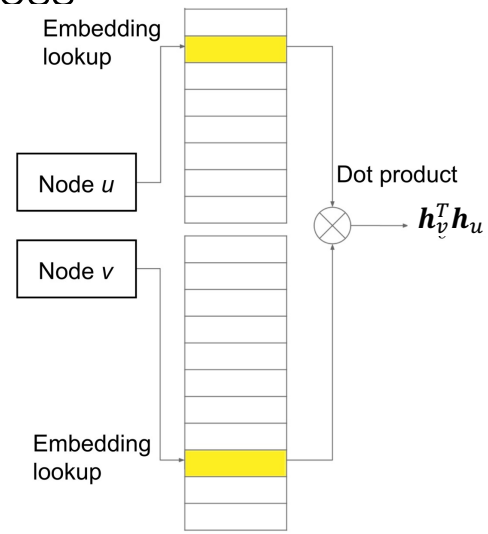
Disease similarity network

2-dimensional node embeddings

Shallow network embeddings

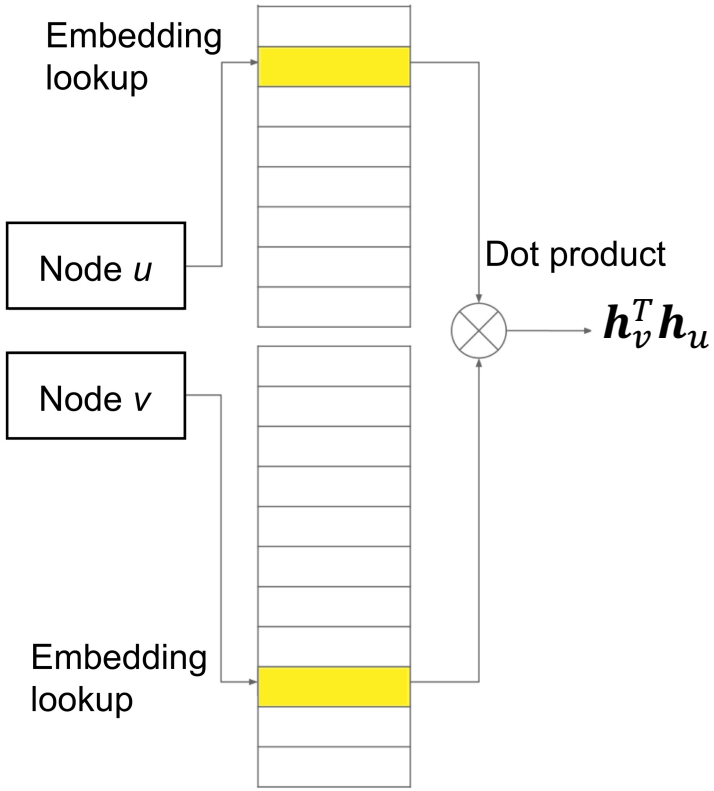
- Goal: Similarity in the embedding space approximates similarity in the network
- Three main stages:
  1. Given a pair of nodes *u*, *v* in network *G*, obtain function *f* to map these nodes to an embedding space to generate $\mathbf{h}_u$ and $\mathbf{h}_v$
  2. Define network similarity $f_n(u,v)$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





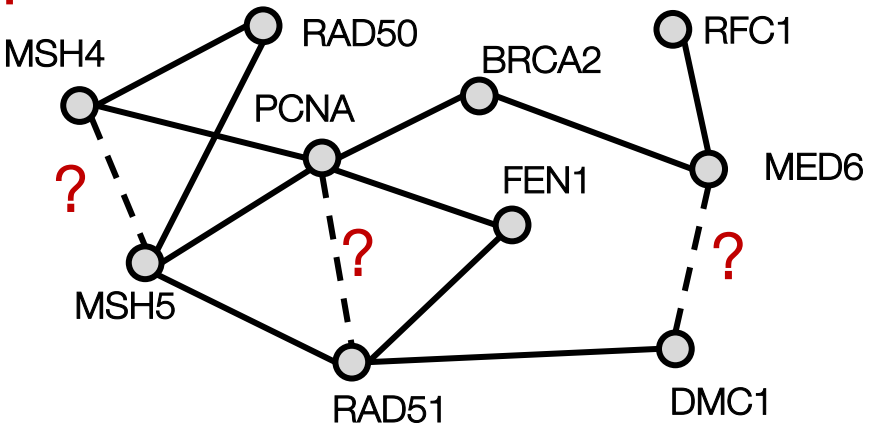
Shallow network embeddings

- Summary:
  - One-layer of data transformation
  - A single hidden layer maps node u to embedding $\boldsymbol{h}_u$ 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)?

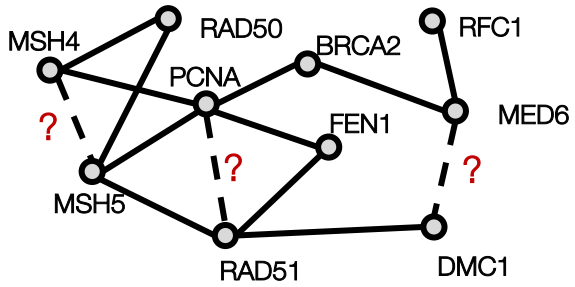
- Given u and v, define an operator g that generates an embedding for pair (u, v):

$$\boldsymbol{h}_{(u,v)} = g(u, v)$$

- Examples of choices for g

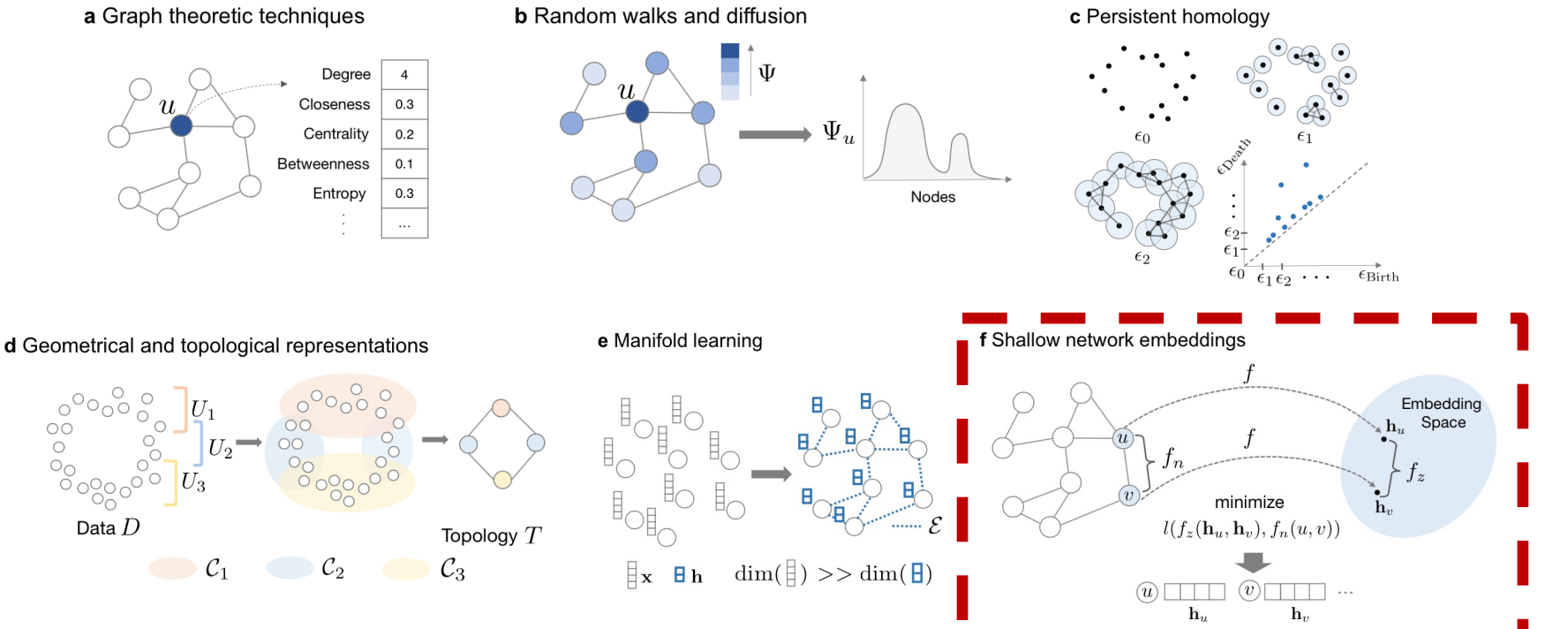
Scoring node pairs	Definition
(a) Average	$[\mathbf{z}_u \boxplus \mathbf{z}_v]_i = \frac{\mathbf{z}_u(i)+\mathbf{z}_v(i)}{2}$
(b) Hadamard	$[\mathbf{z}_u \boxdot \mathbf{z}_v]_i = \mathbf{z}_u(i) * \mathbf{z}_v(i)$
(c) Weighted-L1	$\|\mathbf{z}_u \cdot \mathbf{z}_v\|_{\bar{1}i} = \|\mathbf{z}_u(i) - \mathbf{z}_v(i)\|$
(d) Weighted-L2	$\|\mathbf{z}_u \cdot \mathbf{z}_v\|_{\bar{2}i} = \|\mathbf{z}_u(i) - \mathbf{z}_v(i)\|^2$

Experimental Setup



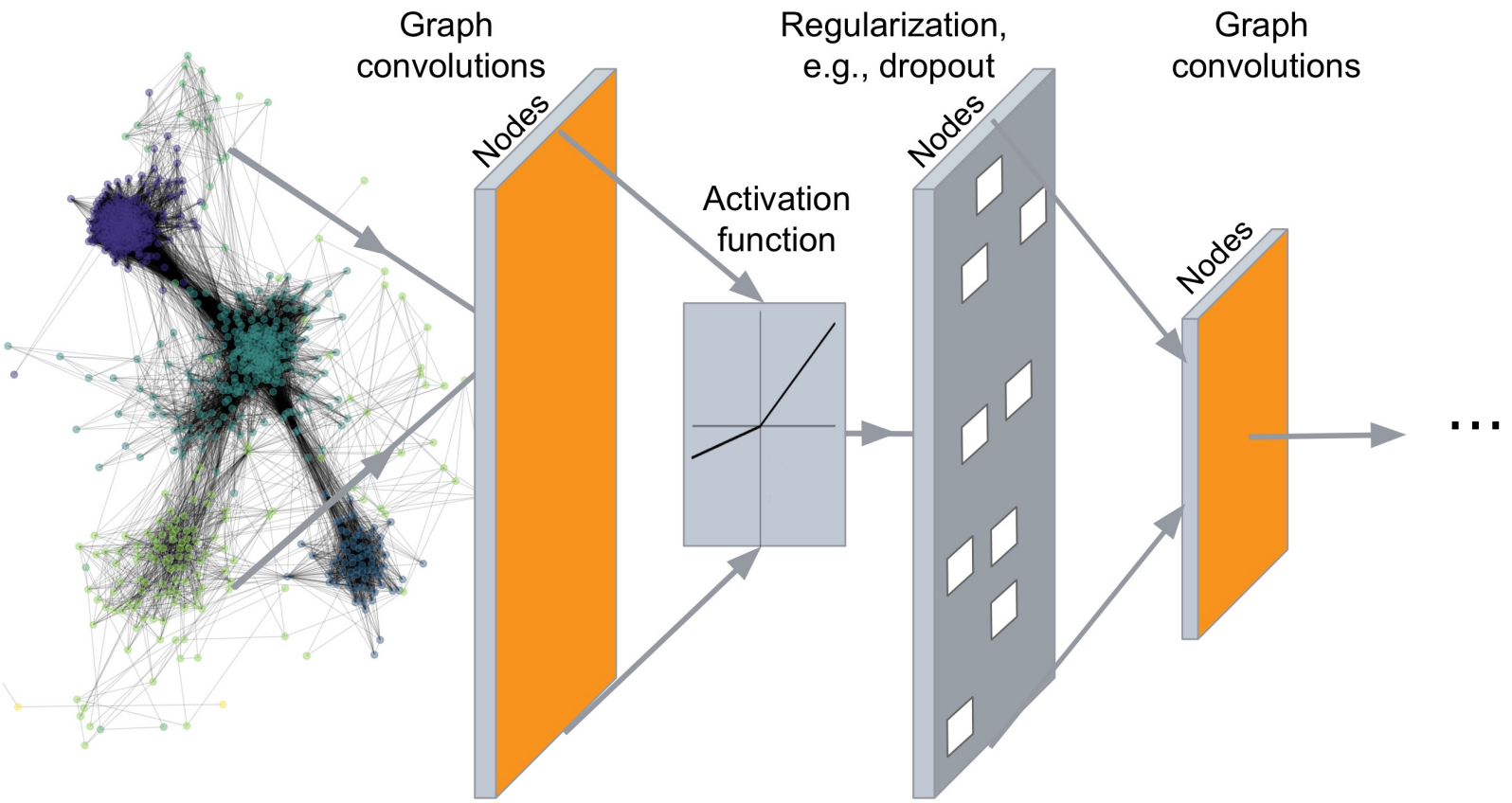
- We are given a PPI network with a certain fraction of edges removed:
  - Remove about 50% of edges
  - Randomly sample an equal number of node pairs that have no edge connecting them
  - Explicitly removed edges and non-existent (or false) edges form a balanced test data set
- Two main stages:
  1. 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



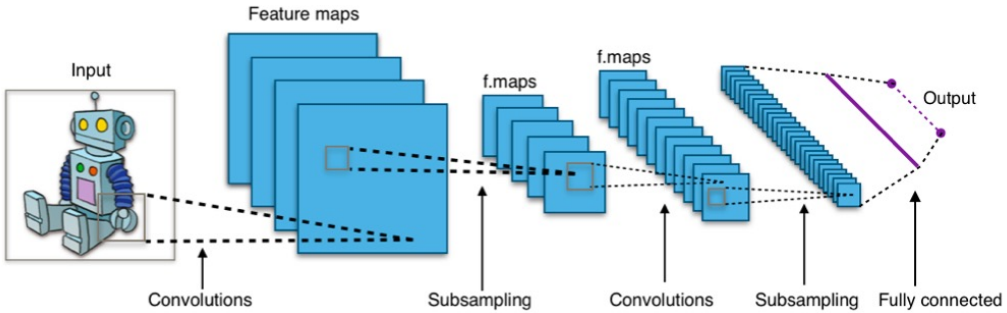
Graph neural networks

- Encoder: Multiple layers of nonlinear transformation of graph structure



Convolutional networks

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



- Single convolutional network with a 3x3 filter:

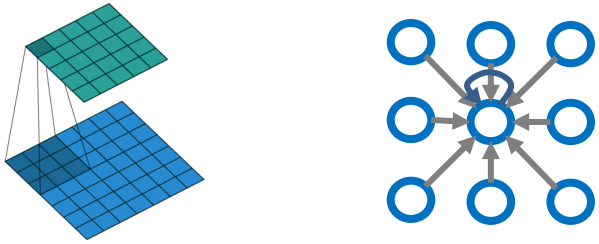
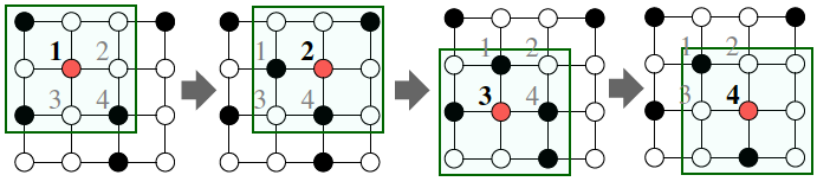


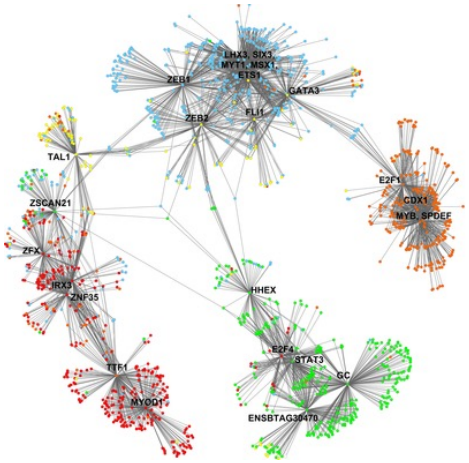
Image Graph

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

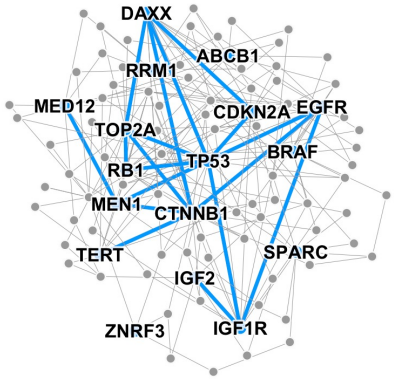


Real world graphs

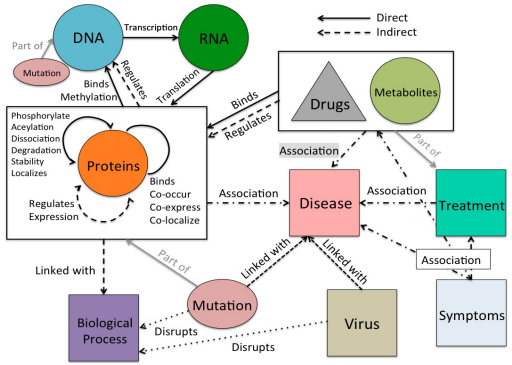
- But what if your graphs look like this?



Gene interaction network



Disease pathways

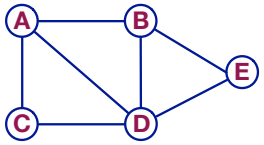


Biomedical knowledge graphs

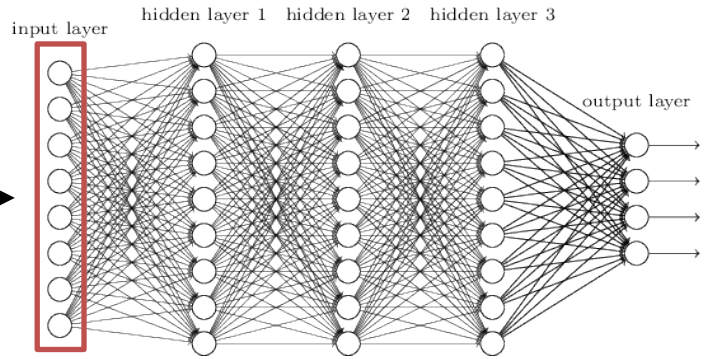
- Examples:
  - Biological or medical networks
  - Social networks
  - Information networks
  - Knowledge graphs
  - Communication networks
  - Web graphs
  - …

Naïve approach

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



	A	B	C	D	E	Feat	
A	0	1	1	1	0	1	0
B	1	0	0	1	1	0	0
C	1	0	0	1	0	0	1
D	1	1	1	0	1	1	1
E	0	1	0	1	0	1	0

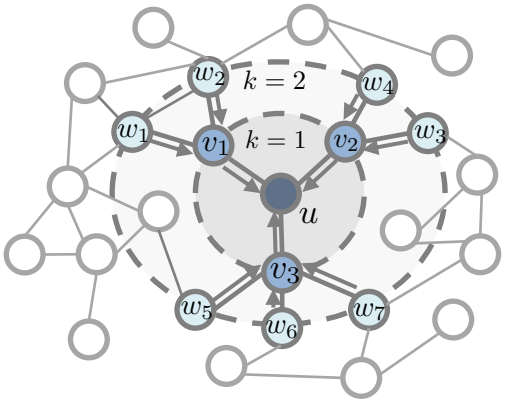


?

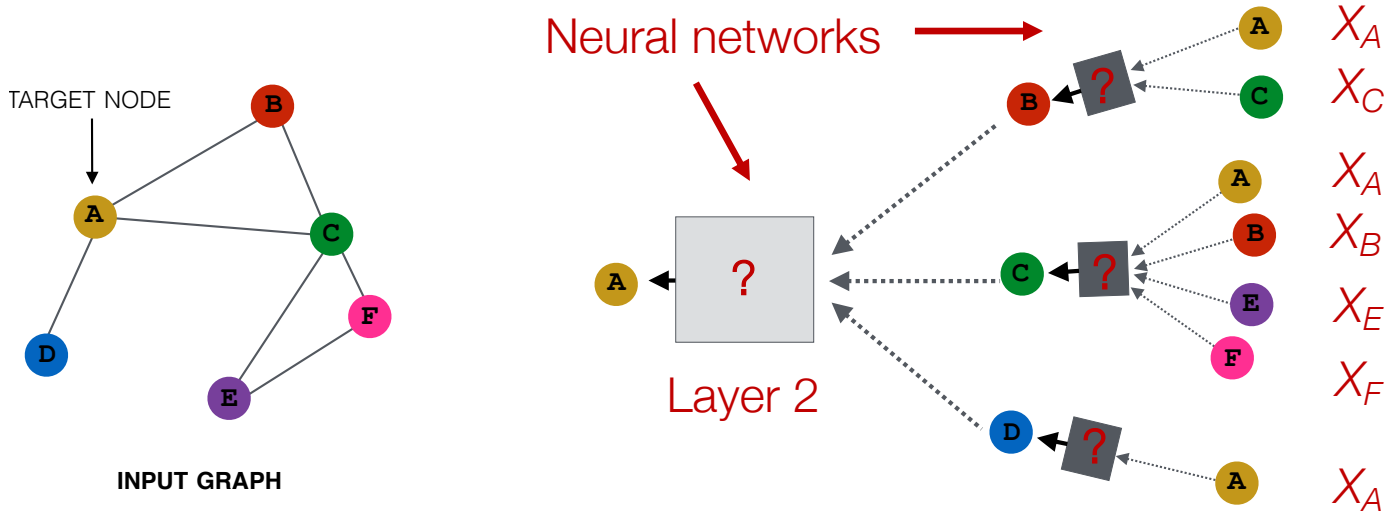
- Issues with this idea:
  - $O(N)$ parameters
  - Not applicable to graphs of different sizes
  - Not invariant to node ordering

Graph neural networks

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

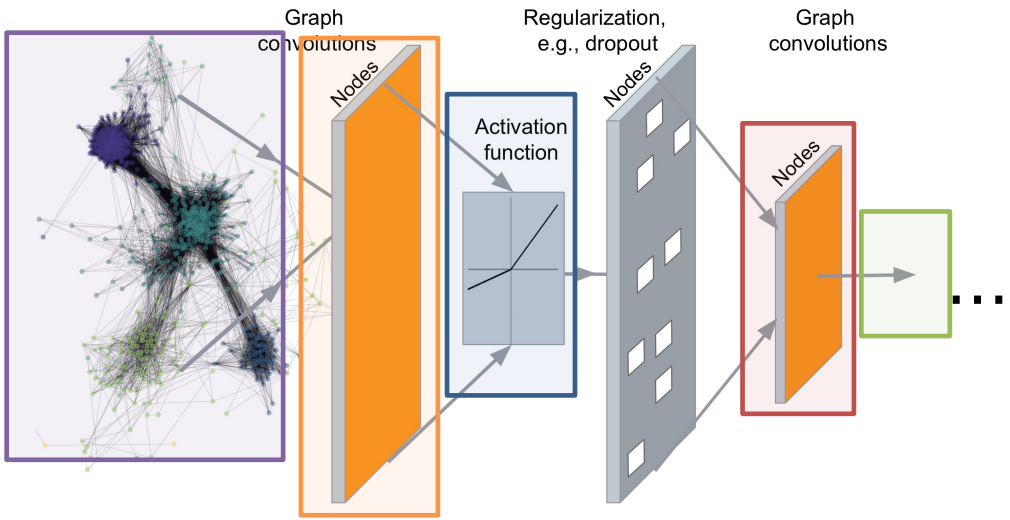


- Neighborhood aggregation:



  - 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. □): Average information from neighbors and apply a neural network

Basic approach



Initial 0-th layer embeddings are equal to node features

$$\mathbf{h}_v^0 = \mathbf{x}_v$$

Previous layer embedding of v

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right), \quad \forall k \in \{1, ..., K\}$$

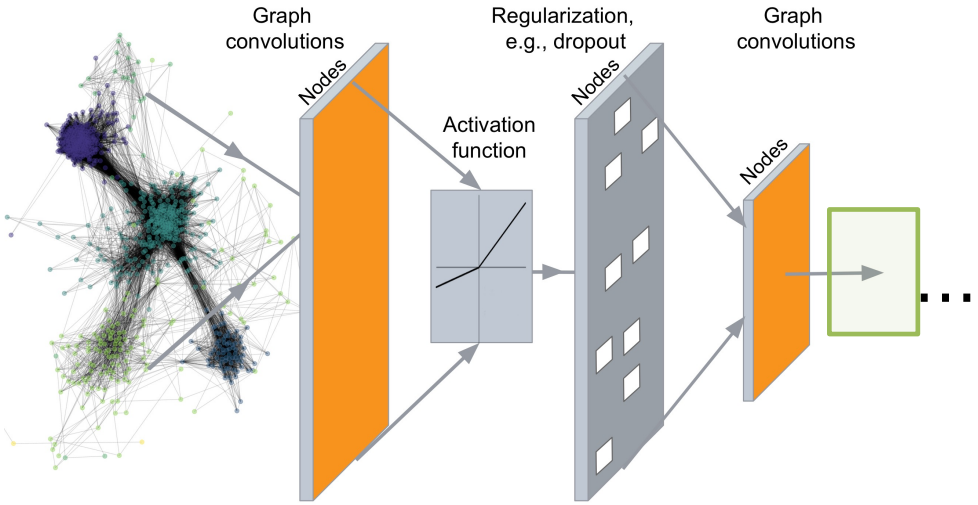
Average of neighbor's previous layer embeddings

$$\mathbf{z}_v = \mathbf{h}_v^K$$

Embedding after K layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

Basic approach



trainable weight matrices (i.e., what we learn)

$$\mathbf{h}_v^0 = \mathbf{x}_v$$

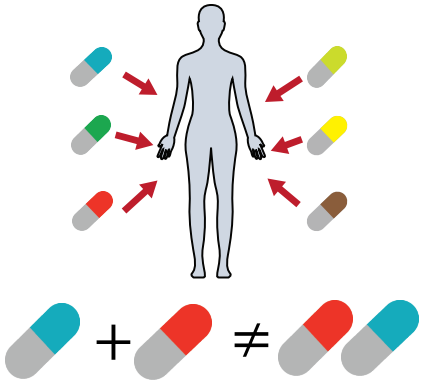
$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right), \quad \forall k \in \{1, ..., K\}$$

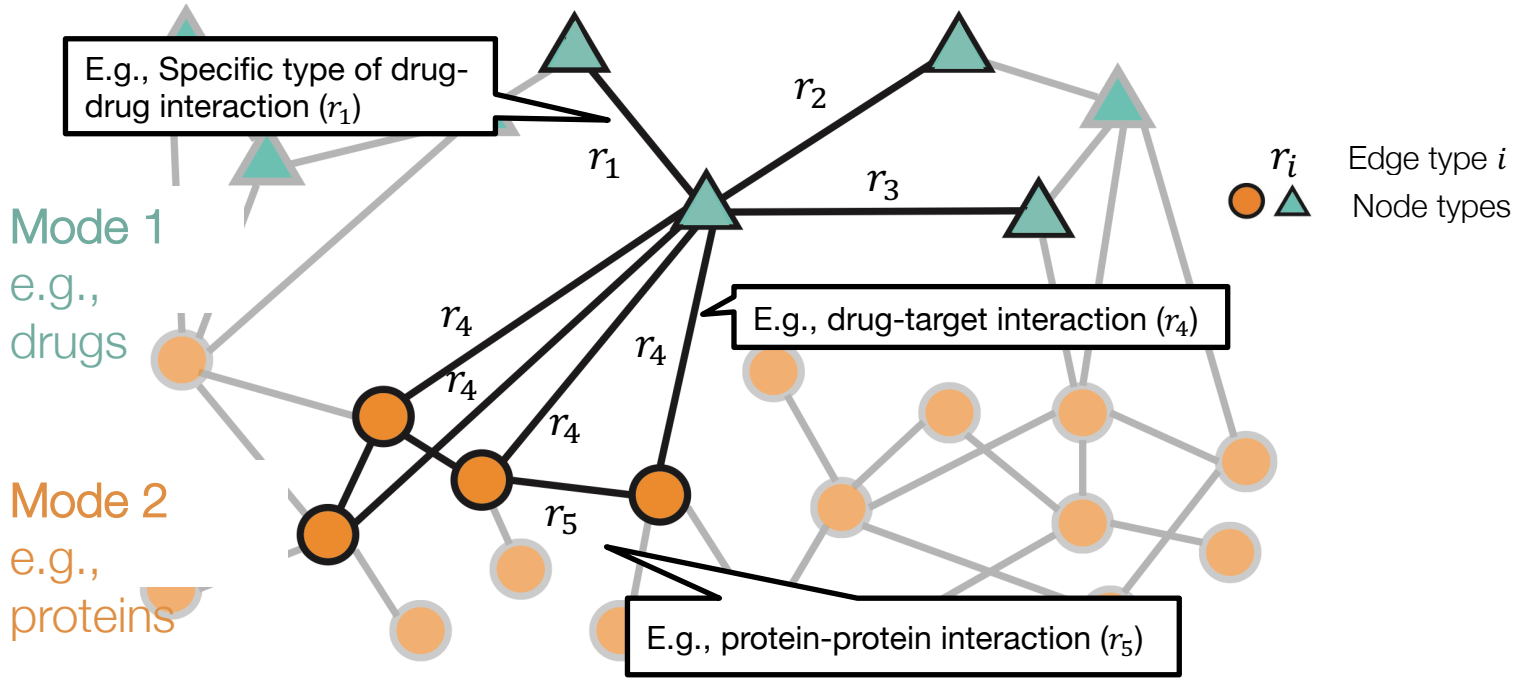
$$\mathbf{z}_v = \mathbf{h}_v^K$$

We can feed these into any loss function and run stochastic gradient descent to train the weight parameters

Application: Prioritize drug combos

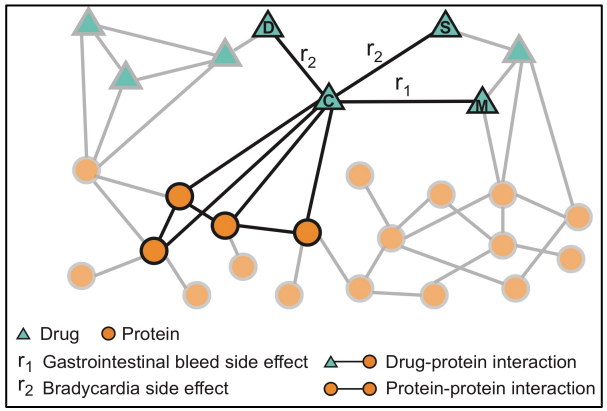
- Combinatorial explosion
  - >13 million possible combinations of 2 drugs
  - >20 billion possible combinations of 3 drugs
- Non-linear & non-additive interactions
  - Different effect than the additive effect of individual drugs
- Small subsets of patients
  - Side effects are interdependent
  - No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





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

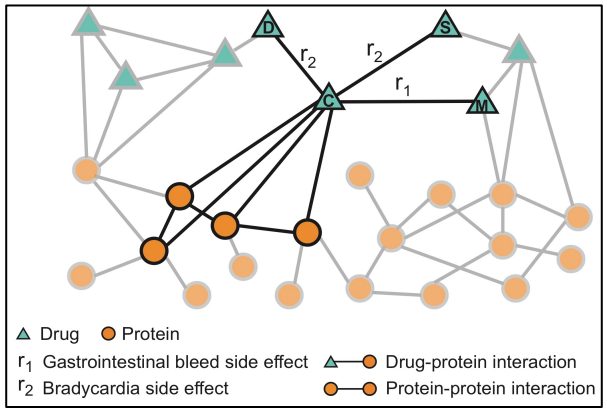
Polypharmacy dataset



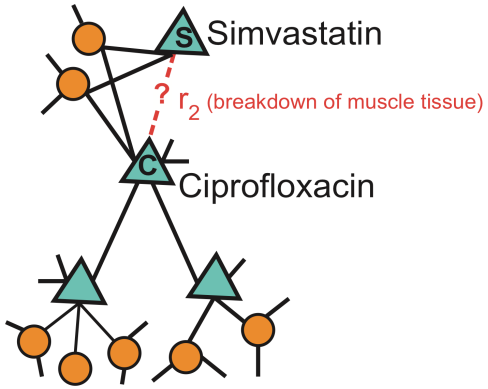
- Molecular, drug, and patient data for all drugs prescribed in US
  - 4,651,131 drug-drug edges: Patient data from adverse event system, tested for confounders [FDA]
  - 18,596 drug-protein edges
  - 719,402 protein-protein edges: Physical, metabolic enzyme-coupled, and signaling interactions
  - Drug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

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

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

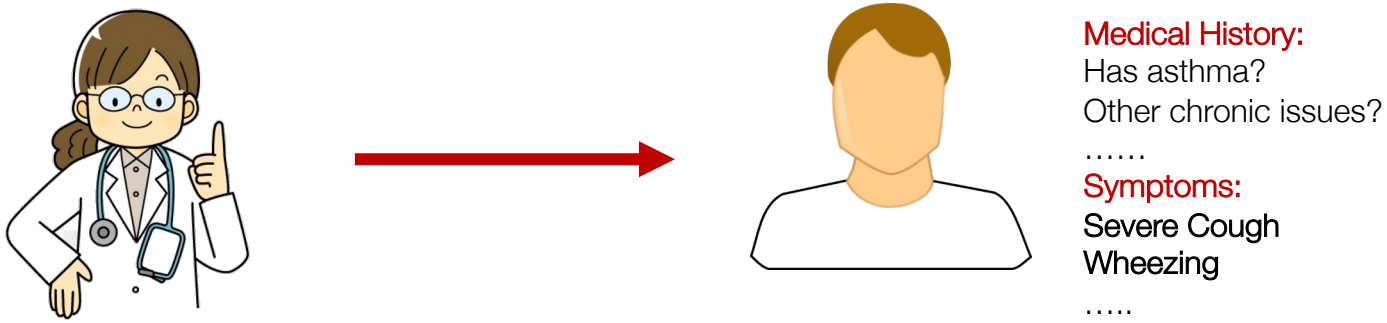


Example: How likely will Simvastatin and Ciprofloxacin, when taken together, break down muscle tissue?

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

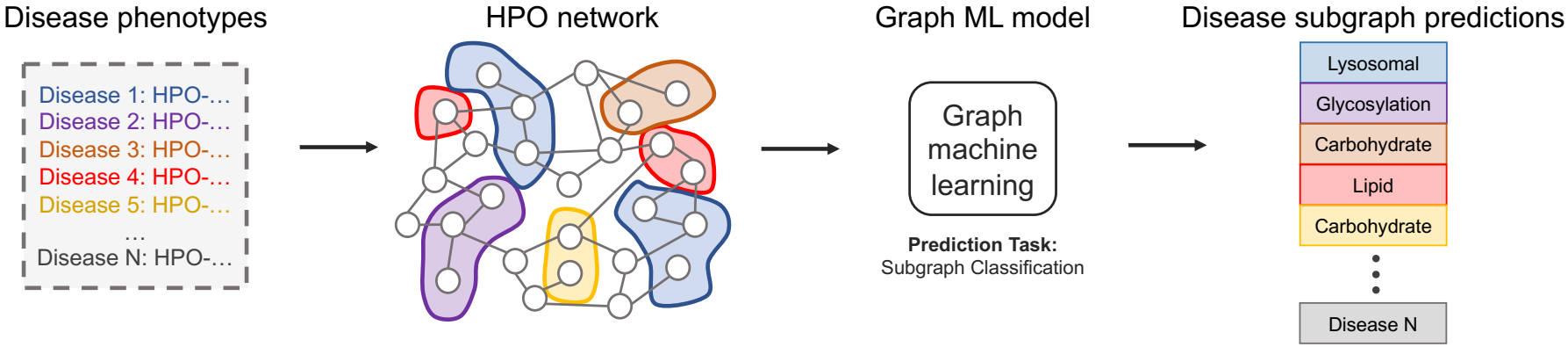
Application: Diagnose patients

- 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



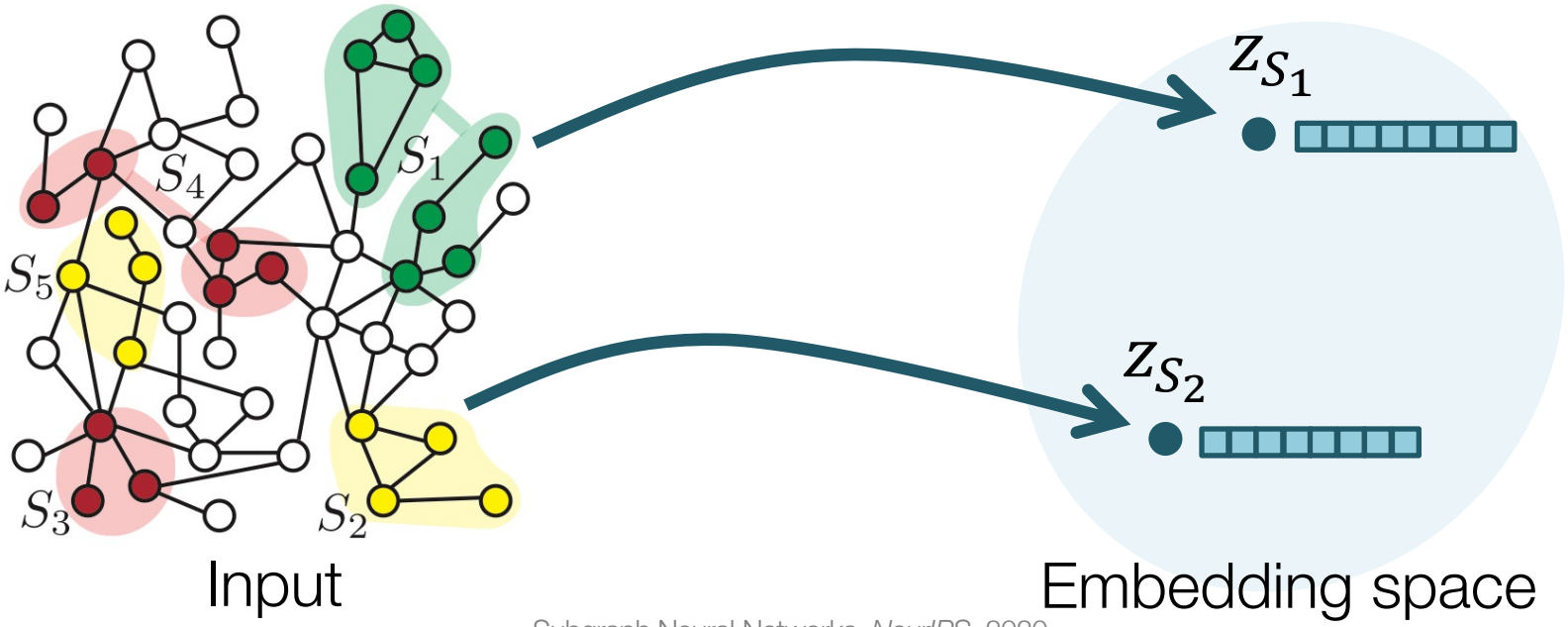
Application: Diagnose patients

- **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

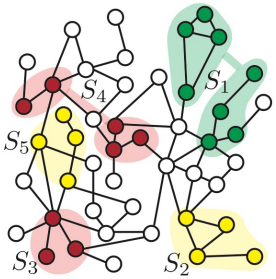
- **Goal:** Learn subgraph embeddings such that the likelihood of preserving subgraph topology is maximized in the embedding space
  - S_i and S_j with similar subgraph topology should be embedded close together in the embedding space

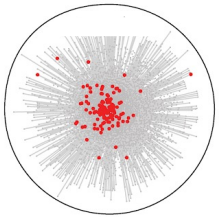


Subgraph Neural Networks, *NeurIPS*, 2020

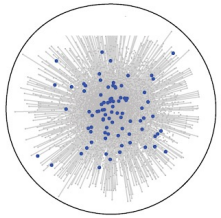
Why are subgraphs challenging?

- 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





Localized

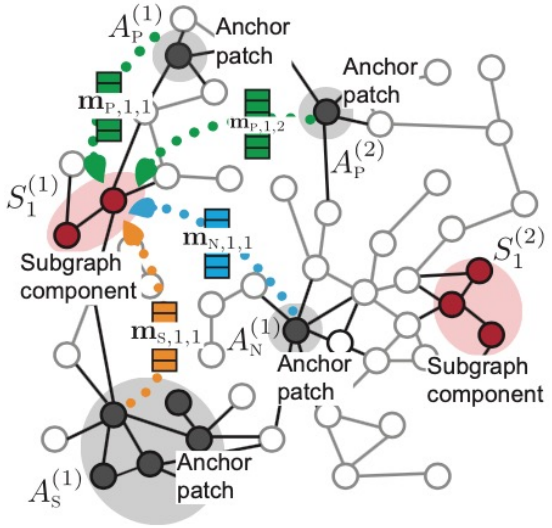


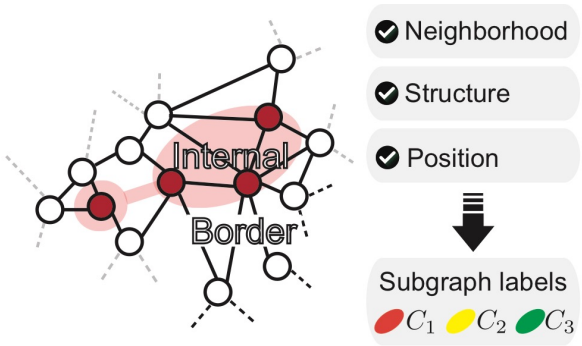
Distributed

Subgraph Neural Networks, *NeurIPS*, 2020

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





Sub-GNN Channel	Sub-GNN Subchannel	
	Internal (I)	Border (B)
Position (P)	Distance between S_i's components	Distance between S_i and rest of G
Neighborhood (N)	Identity of S_i's internal nodes	Identity of S_i's border nodes
Structure (S)	Internal connectivity of S_i	Border connectivity of S_i

Subgraph Neural Networks, *NeurIPS*, 2020

#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

similarity function between subgraph component and an anchor patch

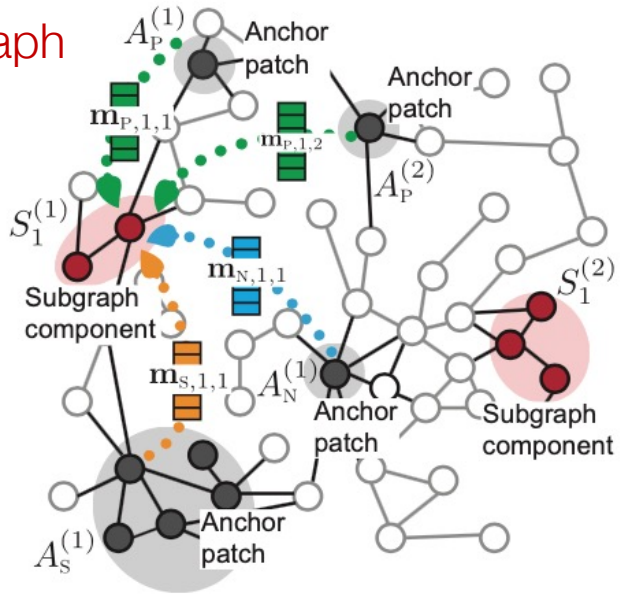
$$\mathbf{MSG}_{\mathrm{X}}^{A\to S} = \gamma_{\mathrm{X}}(S^{(c)}, A_{\mathrm{X}}) \cdot \mathbf{a}_{\mathrm{X}}$$

embedding of anchor patch

$$\mathbf{g}_{\mathrm{X},c} = \mathbf{AGG}_M(\{\mathbf{MSG}_{\mathrm{X}}^{A_{\mathrm{X}}\to S^{(c)}} \; \forall A_{\mathrm{X}} \in \mathcal{A}_{\mathrm{X}}\}),$$

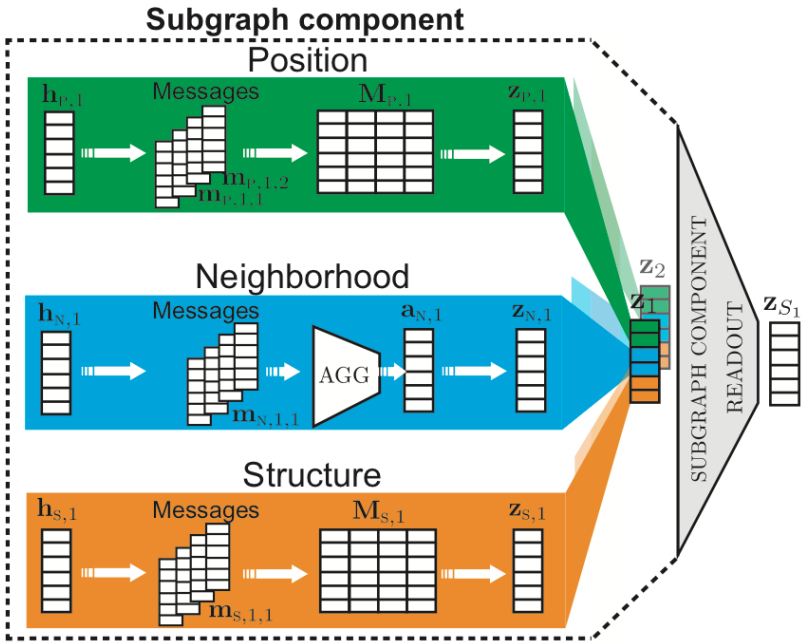
$$\mathbf{h}_{\mathrm{X},c} \leftarrow \sigma(\mathbf{W}_{\mathrm{X}} \cdot [\mathbf{g}_{\mathrm{X},c}; \mathbf{h}_{\mathrm{X},c}]),$$

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



#2: Property-aware routing

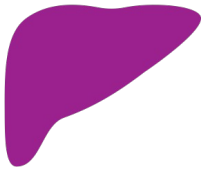
- SubGNN specifies three channels for position, neighborhood, and structure
- Each channel x has three key elements:
  - Similarity function $\gamma_x: \left(S^{(c)}, A_x\right) \rightarrow [0,1]$ to weigh messages exchanged between patches and subgraph components
  - Anchor patch sampling function $\varphi_x: \left(G, S^{(c)}\right) \rightarrow A_x$ 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 $\boldsymbol{z_x}$ are concatenated to produce **a final subgraph representation $\boldsymbol{z_S}$**

Subgraph Neural Networks, *NeurIPS*, 2020

Disease diagnosis datasets



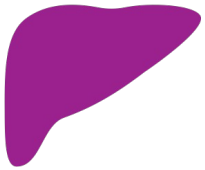
Monogenic Metabolic Diseases
HPO - Orphanet

Monogenic Neurological Diseases
HPO - Orphanet

- 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)?

Subgraph Neural Networks, *NeurIPS*, 2020

Experimental setup



Monogenic Metabolic Diseases
HPO - Orphanet

Monogenic Neurological Diseases
HPO - Orphanet

- 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 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/S01045ED1V01Y202009AIM046)
  - 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)
  - Learning on Graphs Conference (logconference.github.io)

- Tutorials & code bases
  - Pytorch Geometric Colab Notebooks (pytorch-geometric.readthedocs.io/en/latest/notes/colabs.html)
  - Zitnik Lab Graph ML Tutorials (github.com/mims-harvard/graphml-tutorials)
  - Stanford University's CS224 (web.stanford.edu/class/cs224w)

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