BMI 702: Biomedical Artificial Intelligence

Foundations of Biomedical Informatics II, Spring 2023

Lecture 7: Machine learning with heterogeneous graphs, multimodal learning, graph neural networks, knowledge graph embeddings, reasoning over knowledge graphs

HARVARD
MEDICAL SCHOOL

Marinka Zitnik
marinka@hms.harvard.edu

Responses to L6 Quick Check

Examples of networks in biology or medicine. Examples of meaningful predictive or generative tasks

Another example of networks generated in biological sciences are phylogenies. Nodes are organisms at various stages of evolution, and edges (typically also encoding distant ancestry by edge length) are the evolutionary relationships between organisms. Phylogenies are directed networks that demonstrate the divergence of a common ancestor into new species, and they can be used to describe genetic diversity as well as points in which divergence or commonalities between two or more organisms occur.

Microbial network, which represents the relationship and interaction between microbial community. Nodes can represent the microbial species, and edges can be the functional relationship between microbial species such as metabolic interaction. We could use this network to predict the microbial interaction of microbial species by looking at the co-occurrence edges, for example in figuring out positive functional relationship such as mutualism.

Another network example would be using network to model comorbidities. In a comorbidity network, nodes can refer to diseases, and edges refer to co-occurrence of diseases in patients, which can be estimated by statistics such as occurrence frequency. This network can be used for meaningful tasks such as predicting the probability of a certain disease would be involved in a current comorbidity module, analyzing the association between different comorbidity modules based on their connections, and also identifying new comorbidity associations/modules.

Responses to L6 Quick Check

Examples of networks in biology or medicine. Examples of meaningful predictive or generative tasks

A network could be created describing critical care interventions for patients. Nodes would be treatments (i.e. drugs, mechanical ventilation, dialysis). Edges would indicate treatments that are applied together or sequentially for a given patient. Clusters of nodes and edges would be descriptive of illness phenotypes (similar to disease modules for genes/proteins). This may be helpful for anticipating if a patient is already requiring treatment x and treatment y, what is the chance they will also need treatment z, and else are they likely to require?

Another network example I can think of is the medical procedure network. In this network, nodes would represent different medical procedures and edges would represent connection between procedures as well as whether procedure lead to favorable outcome. This type of network would be used to recommend doctors with next procedure steps given current procedure and a model trained on procedure networks of previous patients.

Responses to L6 Quick Check

Examples of networks in biology or medicine. Examples of meaningful predictive or generative tasks

Mobile Health (mHealth) Apps available on common app stores like Google Play Store and Apple App store. Each app is a node. Edges convey similarity of these apps, that is when on the app store page of app A the app store recommends checking out app B (ie lists this app among the "Similar apps" list) then there is a directed node from A to B. A meaningful task on this network is to search for clusters to group mHealth apps into categories, as a basis for more nuanced on diverging trends between clusters.

Nodes: individual patient with autoimmune disease (SSc). Edge: relationship to each other based on distance live from one another using zipcode/area of residence. We could define a predictive task of determining which unlabeled patient would develop interstitial lung disease based on features similar to other patients with SSc in their "neighborhood". This is helpful because this helps us extend exposome data (showing certain zipcodes/neighborhoods have clusters of patients that have the autoimmune disease) to be possibly predictive.

Responses to L6 Quick Check

Additional examples of guilty-by-association approach (i.e., direct neighbor scoring, indirect neighbor scoring, label diffusion)

The guilty-by-association approach can be helpful in predicting the functions of uncharacterized or poorly annotated genes in complex diseases, such as Alzheimer's disease. Alzheimer's disease is a neurodegenerative disorder that affects millions of people worldwide. Despite extensive research efforts, the molecular mechanisms underlying Alzheimer's disease are not fully understood, and many genes involved in the disease process remain poorly characterized. The guilty-by-association approach can be applied to gene expression data from brain tissue samples of Alzheimer's disease patients and healthy controls to predict the functions of uncharacterized genes. By analyzing the network structure of gene co-expression patterns, it is possible to identify genes that are highly connected to known Alzheimer's disease genes, and thus likely to be involved in the disease process. For example, if a poorly annotated gene is highly connected to known Alzheimer's disease genes in the network, it is likely to have a similar function or be involved in the same biological pathway. This information can then be used to guide further experimental validation of the predicted gene functions and potential therapeutic targets for Alzheimer's disease. Overall, the guilty-by-association approach can be a powerful tool for identifying novel genes and pathways involved in complex diseases, and may lead to the development of more effective treatments for these disorders.

Responses to L6 Quick Check

Additional examples of guilty-by-association approach (i.e., direct neighbor scoring, indirect neighbor scoring, label diffusion)

This may be useful in identifying patients who are high users of healthcare who may benefit from more intensive case management or other services. One could identify individuals in the top 5% of health expenditures, and examine features (with a graph network or other mechanism) that are common among them -- i.e. illness phenotypes, geographic location, insurance status, medication use, etc. Then patients with similar features could also be found and assessed for whether they also have high healthcare expenditure.

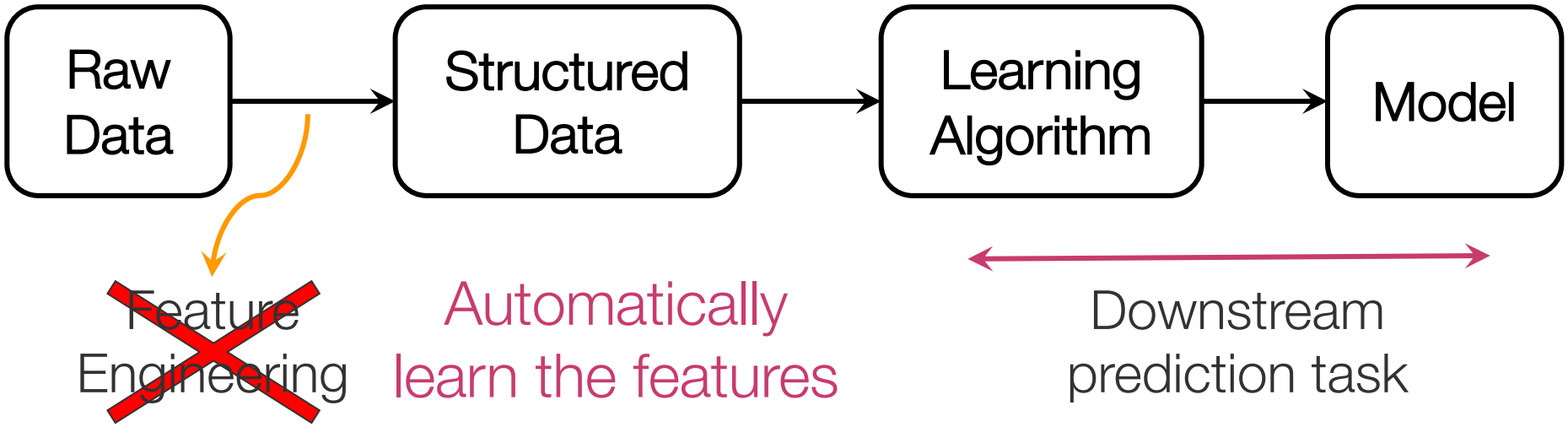
We can use the guilty-by-association approach to predict the efficiency of uncharacterized enzymes at breaking down and neutralizing novel toxins. This involves assigning a score to each uncharacterized enzyme based on its network associations with known enzymes that have similar molecular structures and functions. For example, we can use direct neighbor scoring to assign a score to each uncharacterized enzyme based on the number of known enzymes in its immediate neighborhood that can neutralize the same or similar toxins. Alternatively, we can use indirect neighbor scoring to take into account the functions of the neighbors of the direct neighbors of the uncharacterized enzyme. Finally, we can use label propagation to propagate the labels of known enzymes to uncharacterized enzymes based on their network associations.

Outline for today's lecture

- **Methods:**
  - Shallow graph embeddings
  - Deep graph neural networks
  - Knowledge graph learning
- **Part 1: Applications in biomedical AI**
  - Polypharmacy side effects
  - Antibiotic discovery
- **Part 2: Applications in clinical AI [Michelle M. Li]**
  - Rare genetic disease diagnosis
  - Patient disease state prediction
  - Medication recommendation

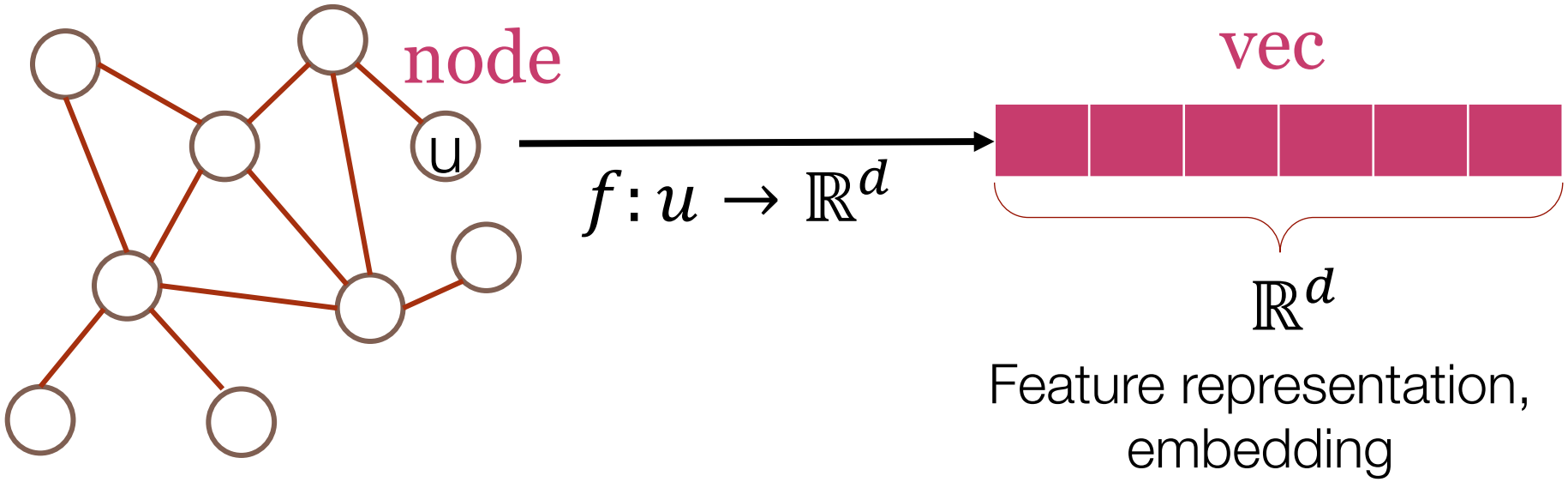
Predictive Modeling Lifecycle

(Supervised) Machine Learning Lifecycle: This feature, that feature. Every single time!

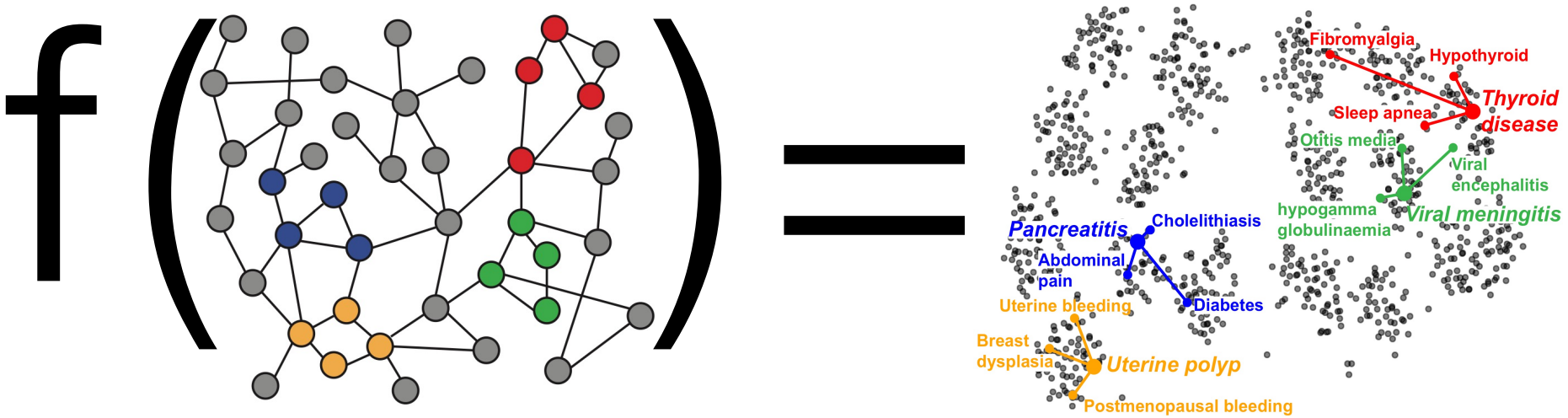


Feature Learning in Graphs

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



Embedding Nodes



Disease similarity network

2-dimensional node embeddings

Input

Output

How to learn mapping function f?

Intuition: Map nodes to embeddings such that similar nodes in the graph are embedded close together

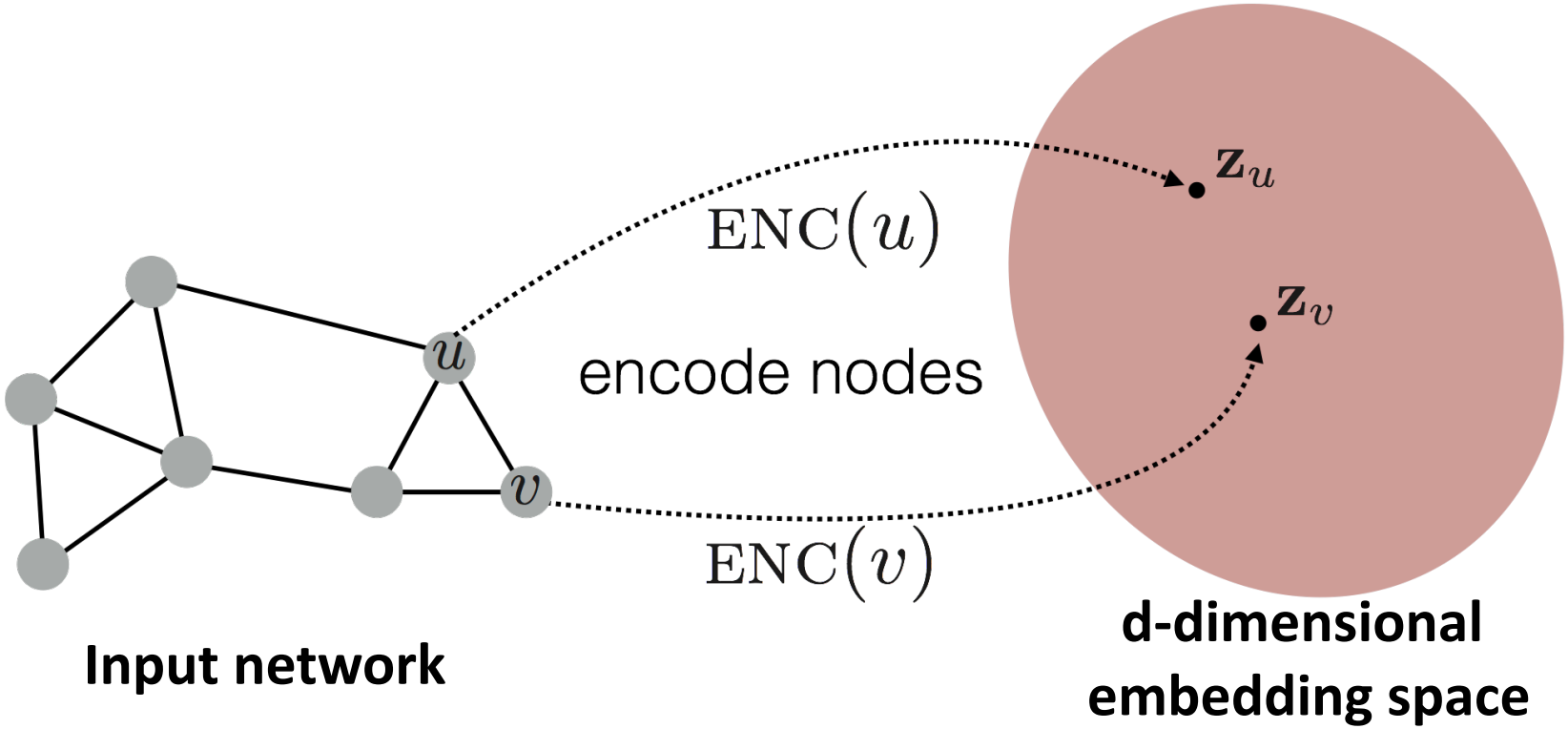
Setup

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

  - **No node features or extra information is used!**

Embedding Nodes

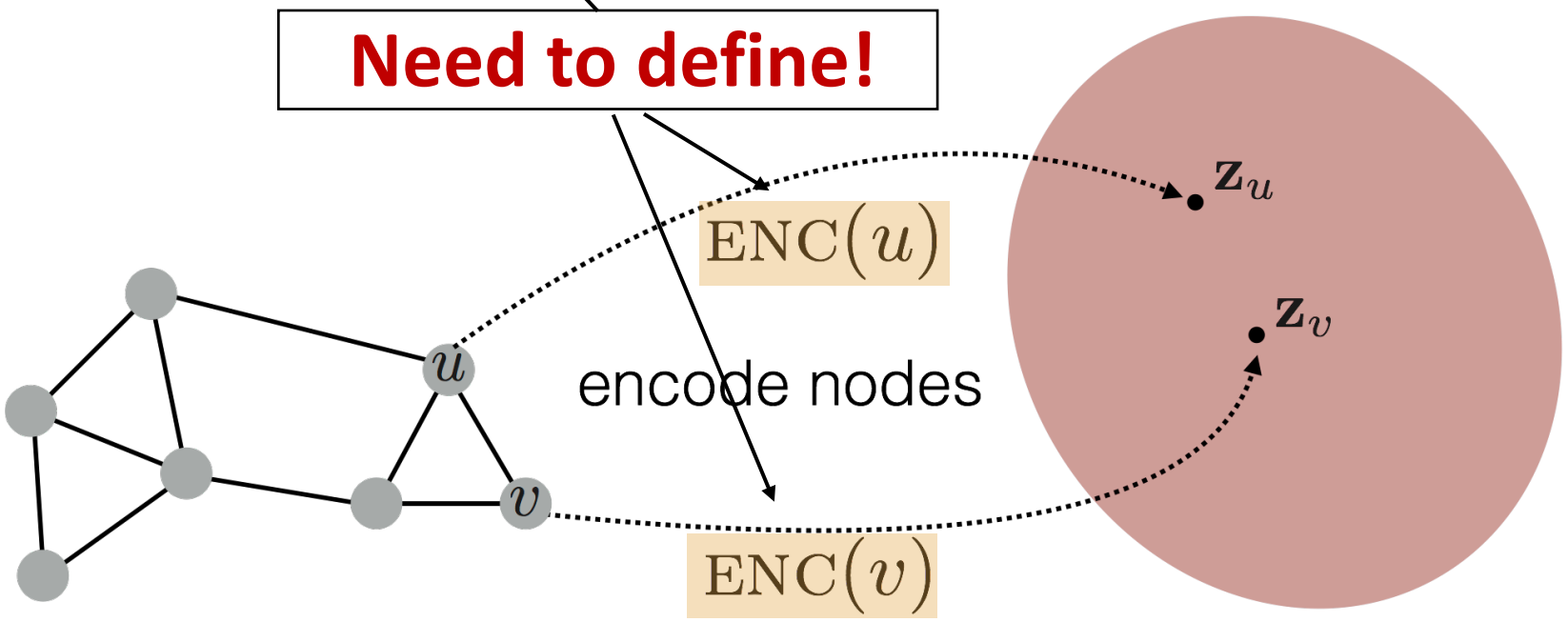
Goal: Map nodes so that similarity in the embedding space (e.g., dot product) approximates similarity in the network



Embedding Nodes

Goal: $\text{similarity}(u, v) \approx \mathbf{z}_v^\top \mathbf{z}_u$

Need to define!



Input network

d-dimensional embedding space

Embedding Nodes: Approach

1. **Define an encoder** (a function ENC that maps node u to embedding $\mathbf{z}_u$)
2. **Define a node similarity function** (a measure of similarity in the input network)
3. **Optimize parameters of the encoder so that:**

$$\text{similarity}(u, v) \approx \mathbf{z}_v^\top \mathbf{z}_u$$

Two Key Components

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

$$\text{ENC}(v) = \mathbf{z}_v$$

$\mathbf{d}$-dimensional embedding

node in the input graph

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

$$\text{similarity}(u, v) \approx \mathbf{z}_v^\top \mathbf{z}_u$$

Similarity of u and v in the network

dot product between node embeddings

Embedding Methods

- Many methods use similar encoders:
  - **Shallow embedders:**
    - node2vec, DeepWalk, LINE, struc2vec
  - **Deep embedders:**
    - Graph neural networks

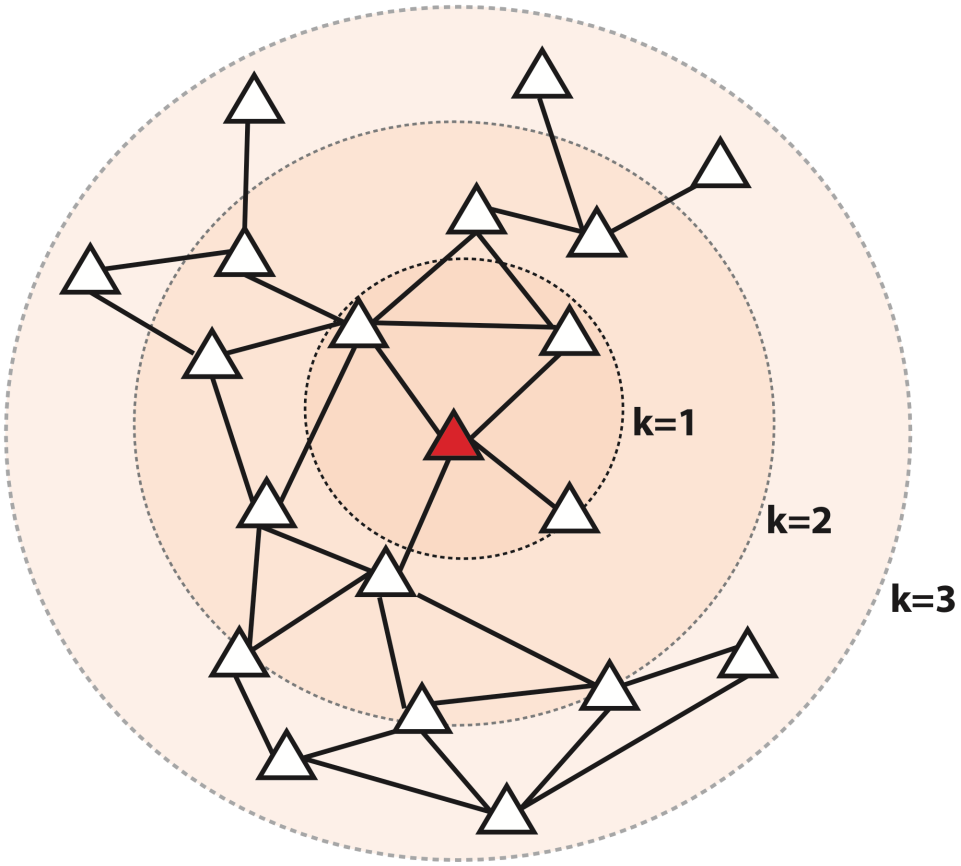
- These methods use different notions of node similarity:
  - Two nodes have similar embeddings if:
    - they are connected?
    - they share many neighbors?
    - they have similar local network structure?
    - etc.

Shallow graph learning

Node2vec: Scalable Feature Learning for Networks

Multi-Hop 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?

Learning Embeddings: Optimization

- Given $G = (V, E)$
- Goal is to learn $f: u \rightarrow \mathbb{R}^d$
  - where f is a table lookup
    - We directly "learn" coordinates $\boldsymbol{z_u} = f(u)$ of u
- Given node u, we want to learn embedding $f(u)$ that is predictive of nodes in u's neighborhood $N_{\mathrm{R}}(u)$:

$$\max_f \sum_{u \in V} \log \Pr(N_{\mathrm{R}}(u) | \boldsymbol{z}_{\mathrm{u}})$$

Learning Embeddings: Optimization

Goal: Find embedding $\mathbf{z}_u$ that predicts nearby nodes $N_R(u)$:

$$\sum_{v \in V} \log(P(N_R(u)|\mathbf{z}_u))$$

Assume conditional likelihood factorizes:

$$P(N_R(u)|\mathbf{z}_u) = \prod_{n_i \in N_R(u)} P(n_i|\mathbf{z}_u)$$

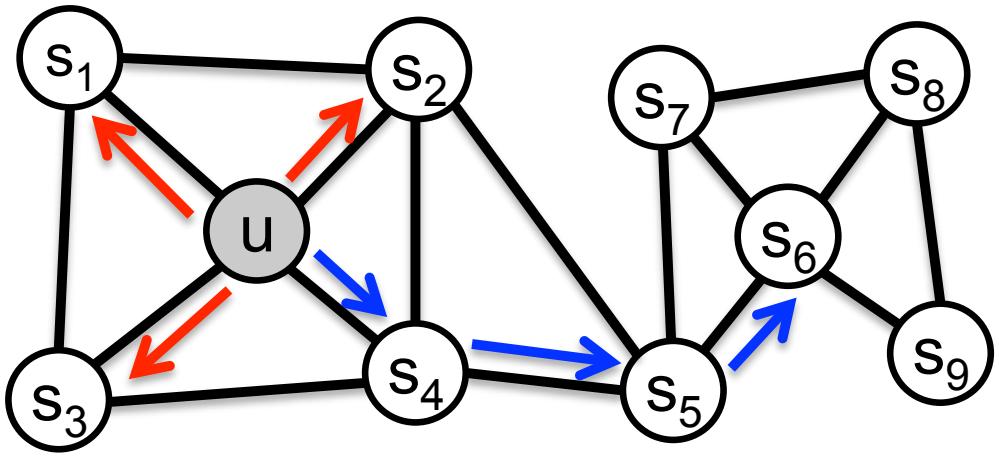
Random-Walk Embeddings

$$\mathbf{z}_u^\top \mathbf{z}_v \approx$$

Probability that u and v 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 $\mathrm{N}_R(u)$ 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 $\mathrm{N}_R(u)$:

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

Random-Walk Optimization

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

sum over all nodes u

sum over nodes v seen on random walks starting from u

predicted probability of u and $\mathbf{v}$ co-occurring on random walk, i.e., use softmax to parameterize $P(v|\mathbf{z}_u)$

Random walk embeddings = $\mathbf{z}_u$ minimizing $\mathcal{L}$

Details

Random Walks: Overview

1. Simulate many short random walks starting from each node using a strategy R
2. For each node u, get $\mathrm{N}_R(u)$ 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 $\mathrm{N}_R(u)$:

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

Can efficiently approximate using negative sampling

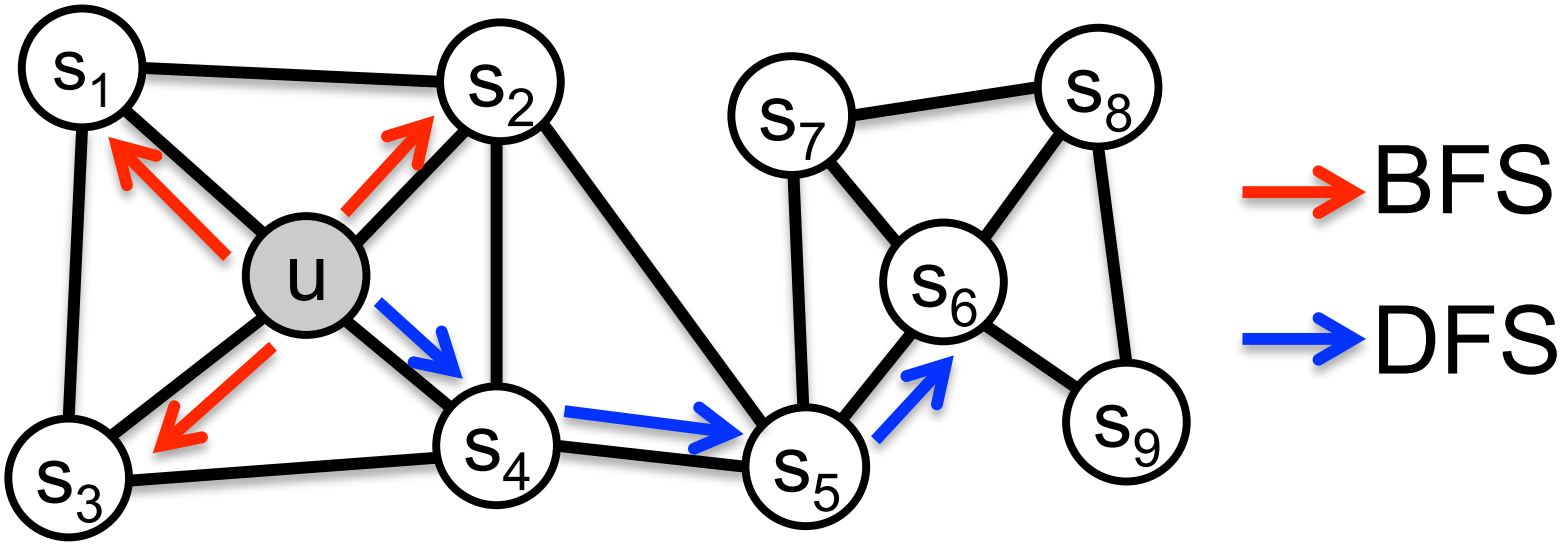
What is the strategy R?

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

Details

node2vec: Biased Walks

Idea: Use biased random walks that can trade off between **local** and **global** views of the network



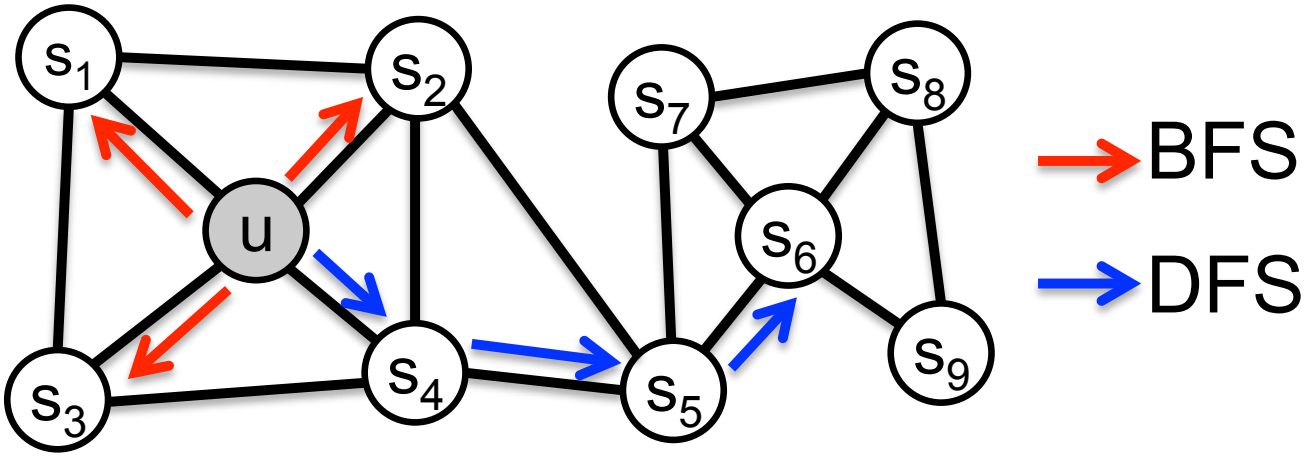
$N_{BFS}(u) = \{s_1, s_2, s_3\}$ **Local** microscopic view

$N_{DFS}(u) = \{s_4, s_5, s_6\}$ **Global** macroscopic view

Details

node2vec: Biased Walks

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



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

$N_{DFS}(u) = \{s_4, s_5, s_6\}$ Global macroscopic view

Details

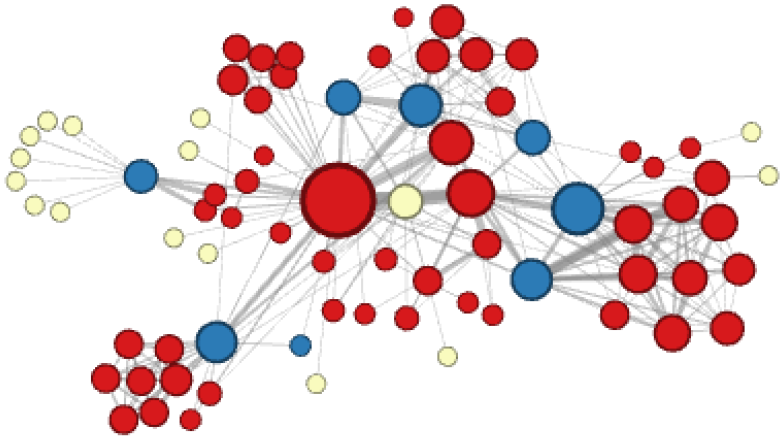
Experiment: Micro vs. Macro

Regions of the network (i.e., communities) are colored using the same color. In this setting node2vec discovers **clusters/communities** of characters that frequently interact with each other in the major sub-plots of the novel

Clusters correspond to **structural roles**. Blue nodes: characters that act as bridges between sub-plots of the novel. Yellow nodes: characters at the periphery with limited interactions

p=1, q=0.5

Microscopic view of the network neighborhood



p=1, q=2

Macroscopic view of the network neighborhood

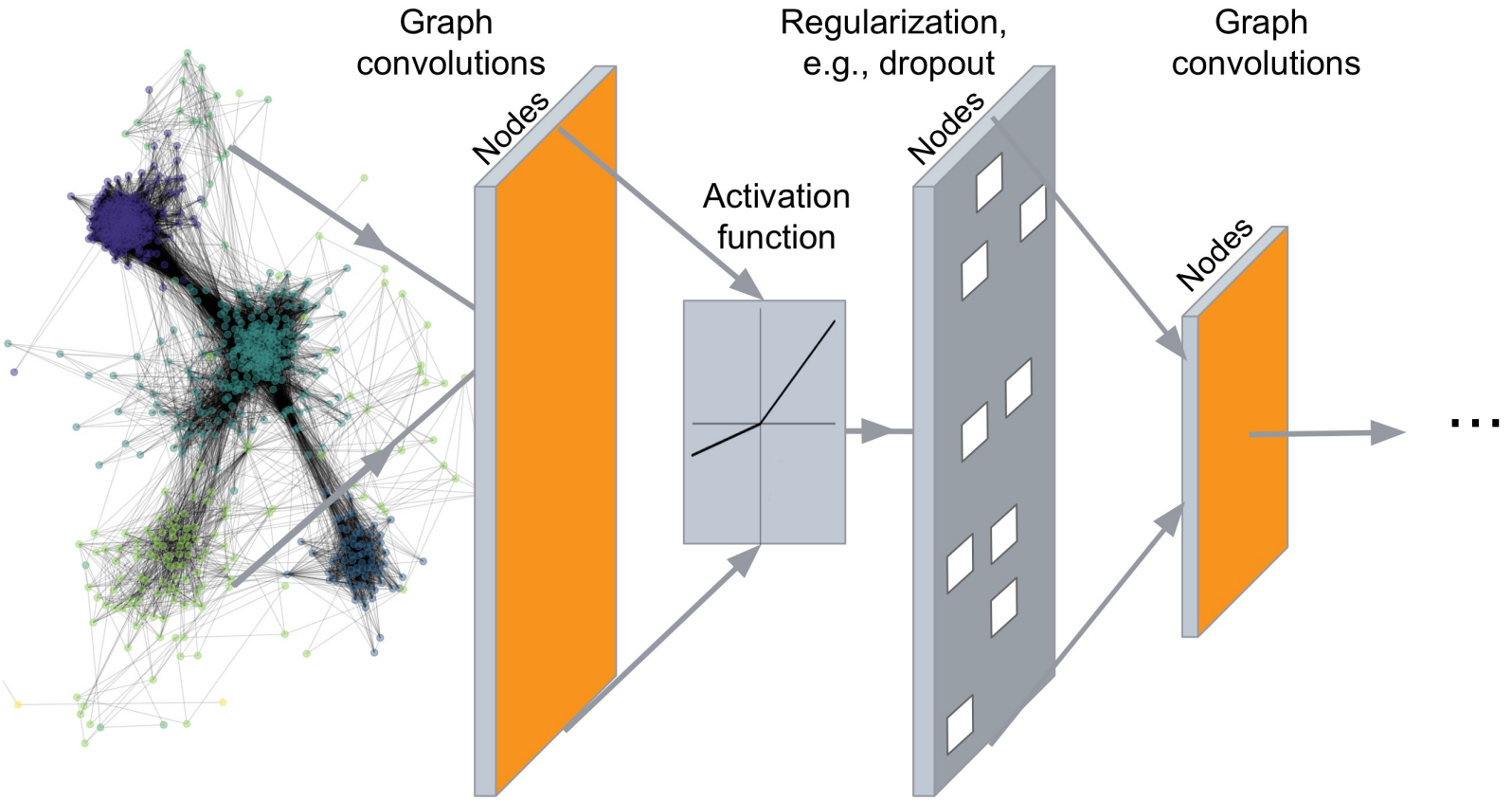
Return parameter p: Return back to the previous node

In-out parameter q: Moving outwards (DFS) vs. inwards (BFS)

Deep graph learning
Graph Neural Networks

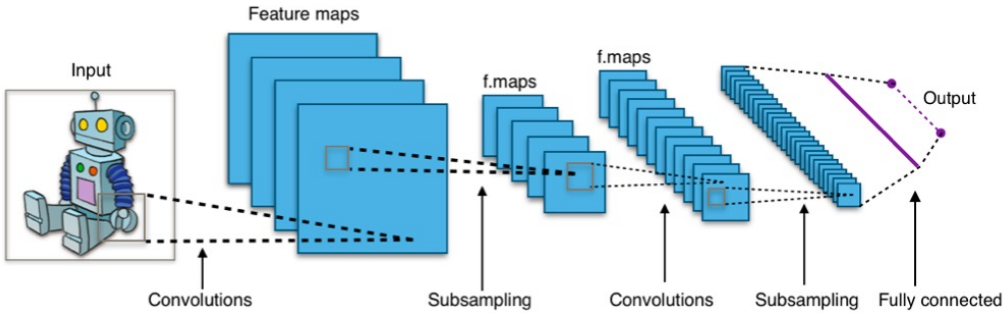
Graph neural networks

- 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:

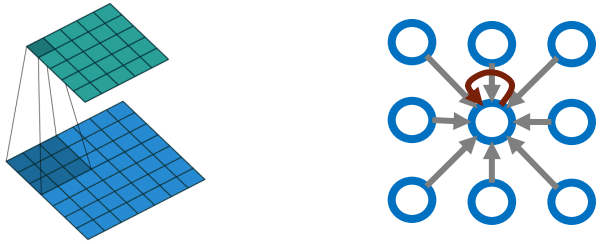
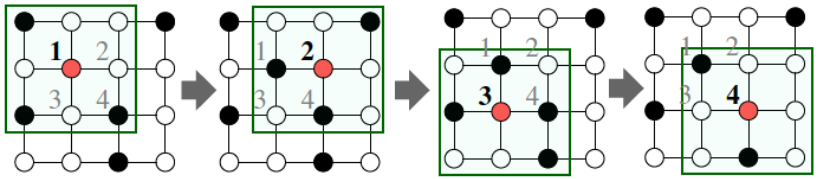


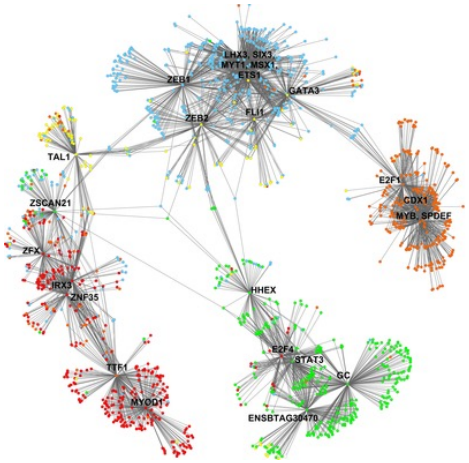
Image Graph

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

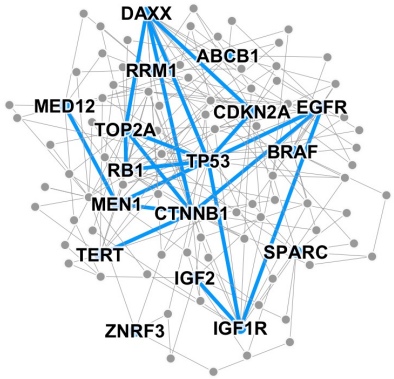


Real world graphs

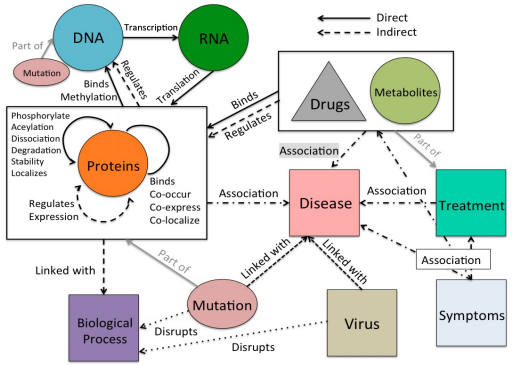
- But what if your graphs look like this?



Gene interaction network



Disease pathways

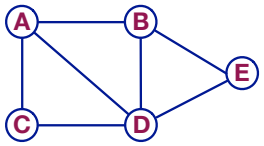


Biomedical knowledge graphs

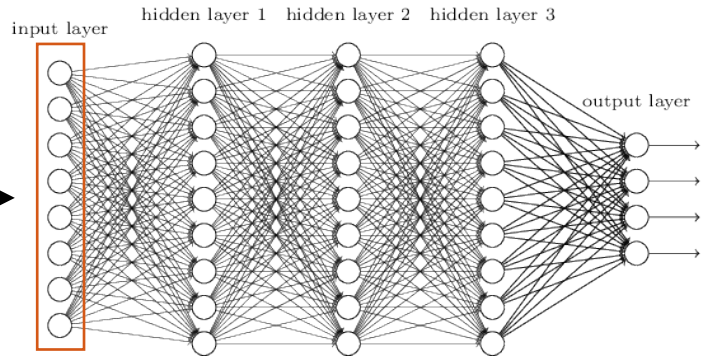
- Examples:
  - Biological or medical networks
  - Social networks
  - Information networks
  - Knowledge graphs
  - Communication networks
  - Web graphs
  - …

Naïve approach

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



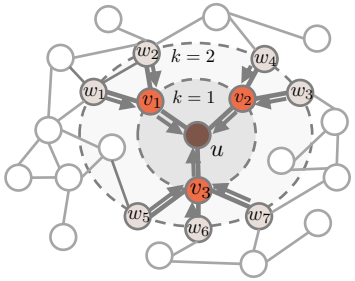
	A	B	C	D	E	Feat	
A	0	1	1	1	0	1	0
B	1	0	0	1	1	0	0
C	1	0	0	1	0	0	1
D	1	1	1	0	1	1	1
E	0	1	0	1	0	1	0



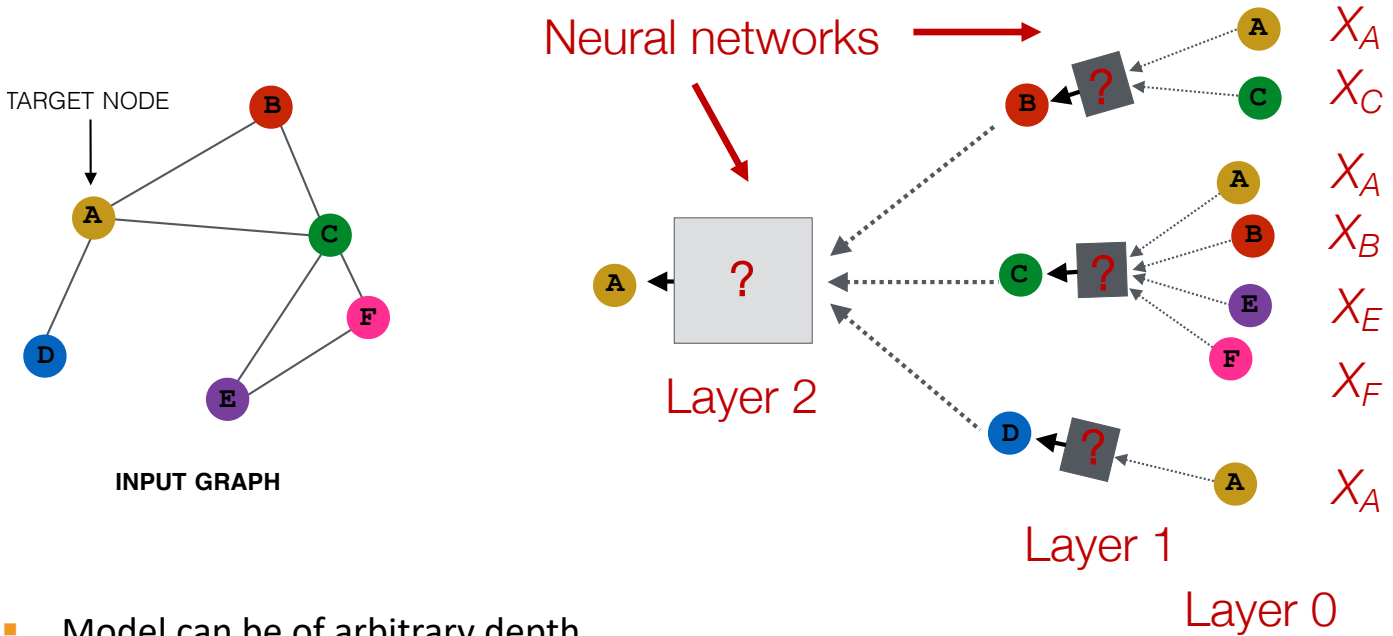
?

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

Graph neural networks

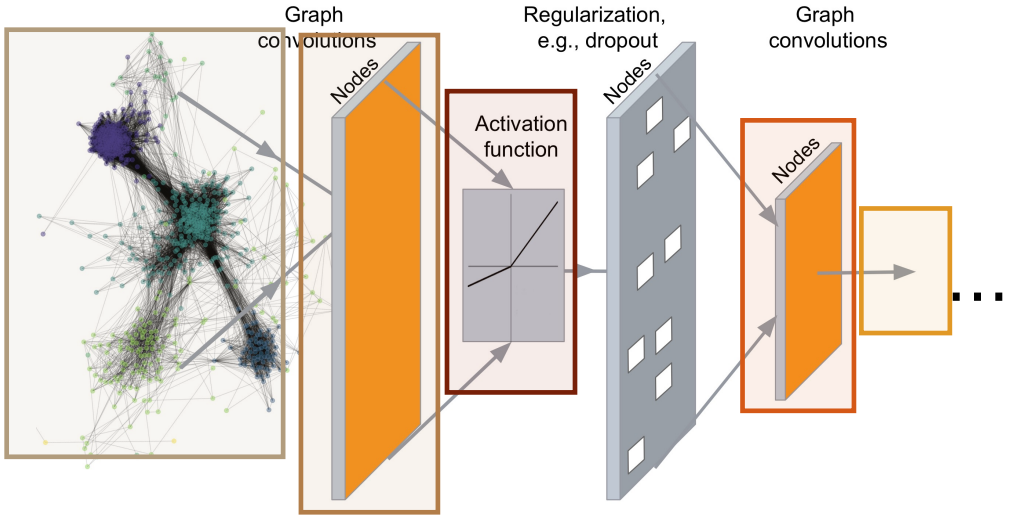


- Intuition:
  - Each node's neighborhood defines a computational graph
  - Generate node embeddings based on local network neighborhoods
- Neighborhood aggregation:



  - 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: 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

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

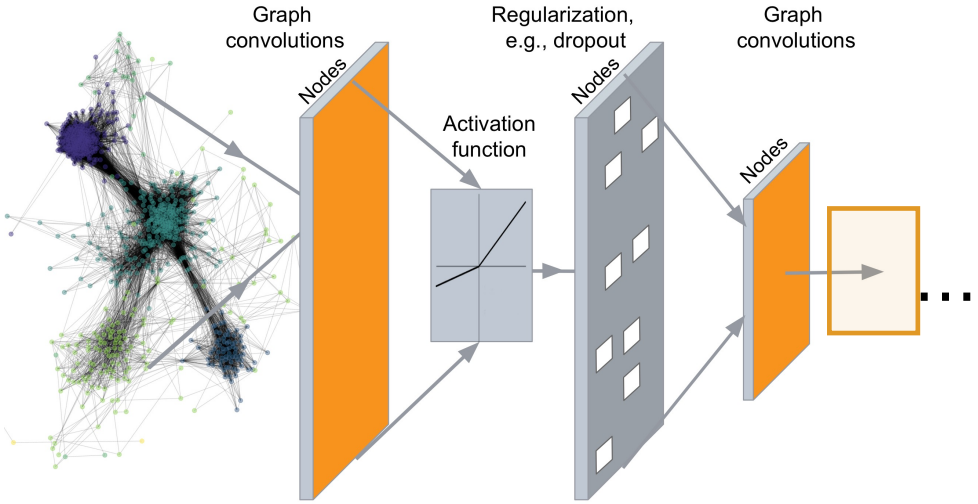
$$\mathbf{z}_v = \mathbf{h}_v^K$$

Average of neighbor's previous layer embeddings

Embedding after K layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

Basic approach



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

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

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

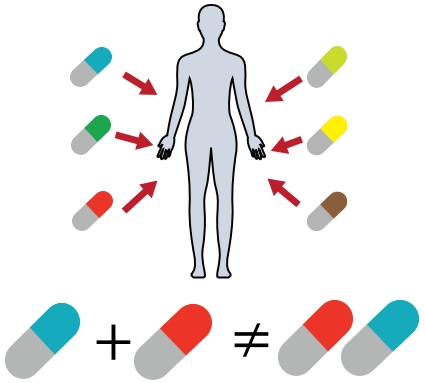
$$\mathbf{z}_v = \mathbf{h}_v^K$$

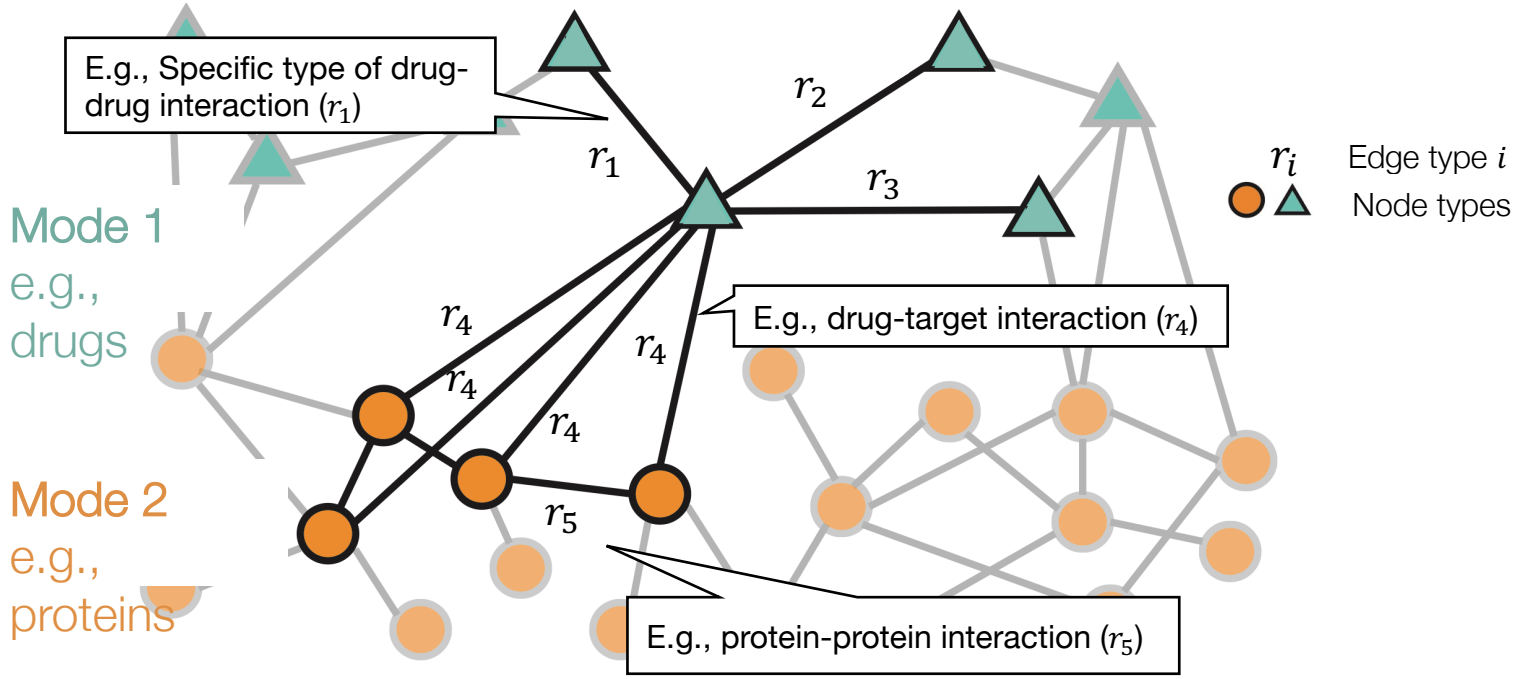
We can feed these into any loss function and run stochastic gradient descent to train the weight parameters

Applications in biomedical AI

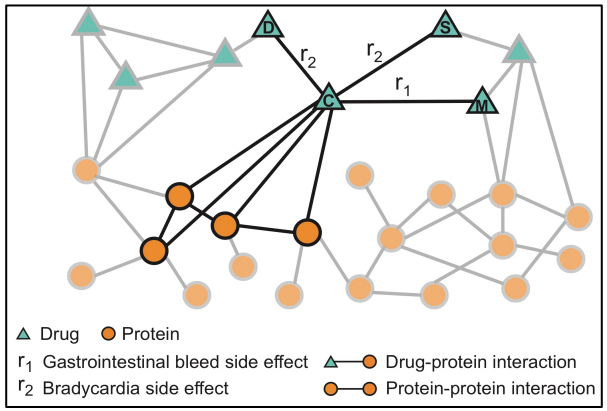
Application: Drug combinations

- 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



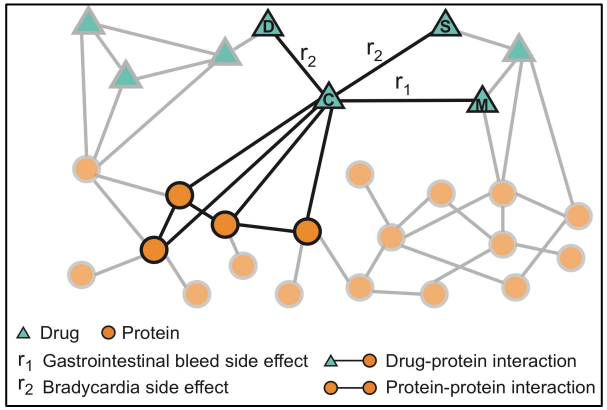


Polypharmacy dataset

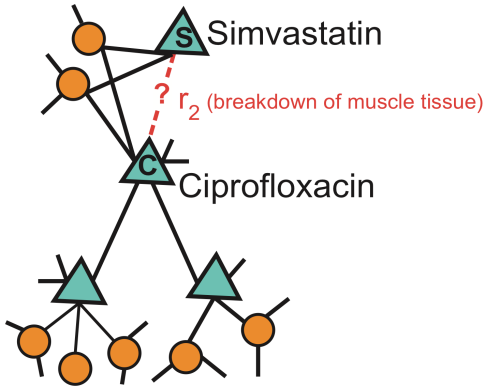


- Molecular, drug, and patient data for all US-approved drugs
  - 4,651,131 drug-drug edges: Patient data from adverse event system, tested for confounders [FDA]
  - 18,596 drug-protein edges
  - 719,402 protein-protein edges: Physical, metabolic enzyme-coupled, and signaling interactions
  - Drug and protein features: drugs' chemical structure, proteins' membership in pathways
- This is a multimodal network with over 5 million edges separated into 1,000 different edge types

Experimental setup

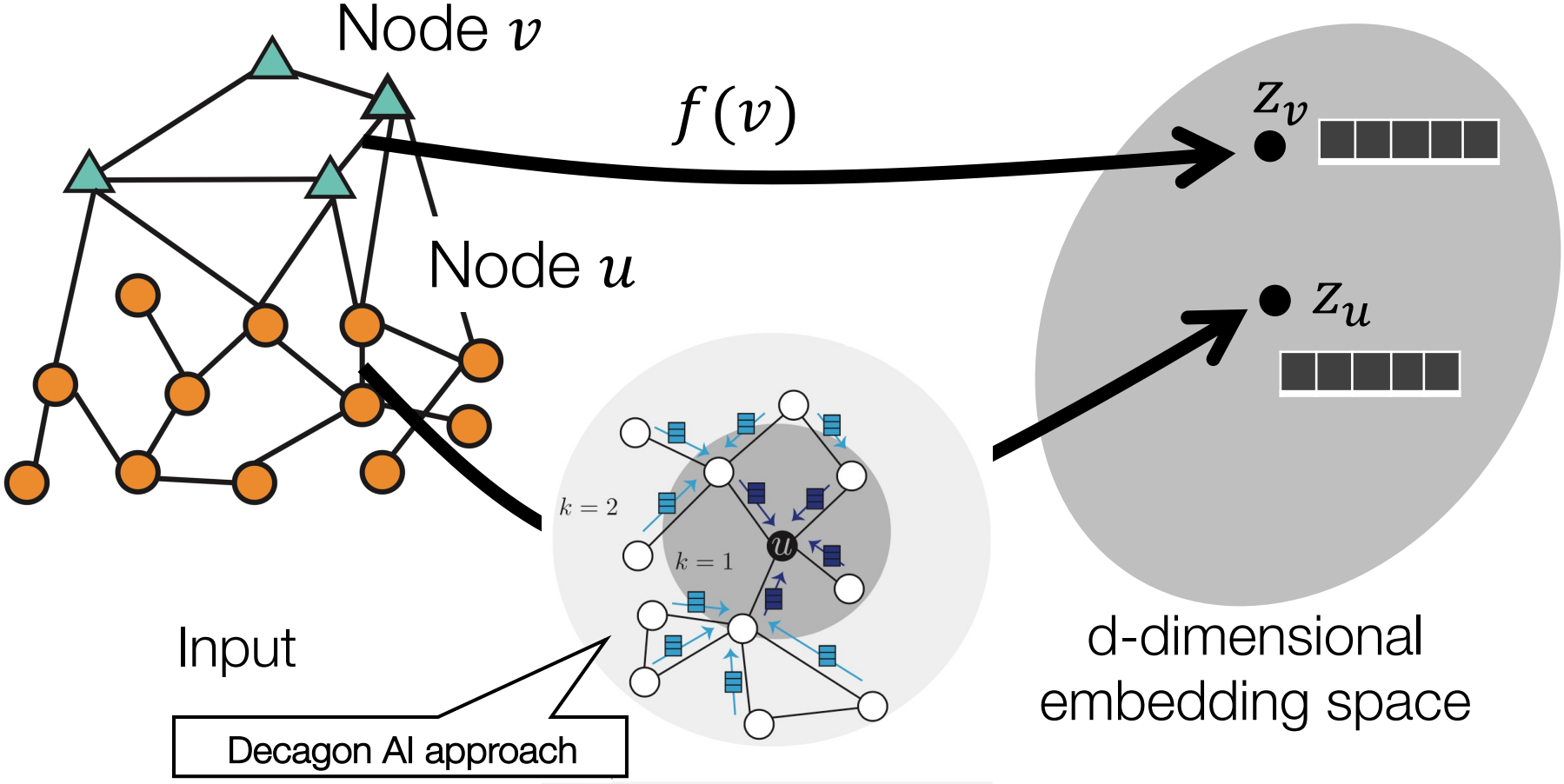


- Two main stages:
  1. Learn an embedding for every node in polypharmacy network
  2. Predict a score for every drug-drug, drug-protein, protein-protein pair in the test set based on the embeddings



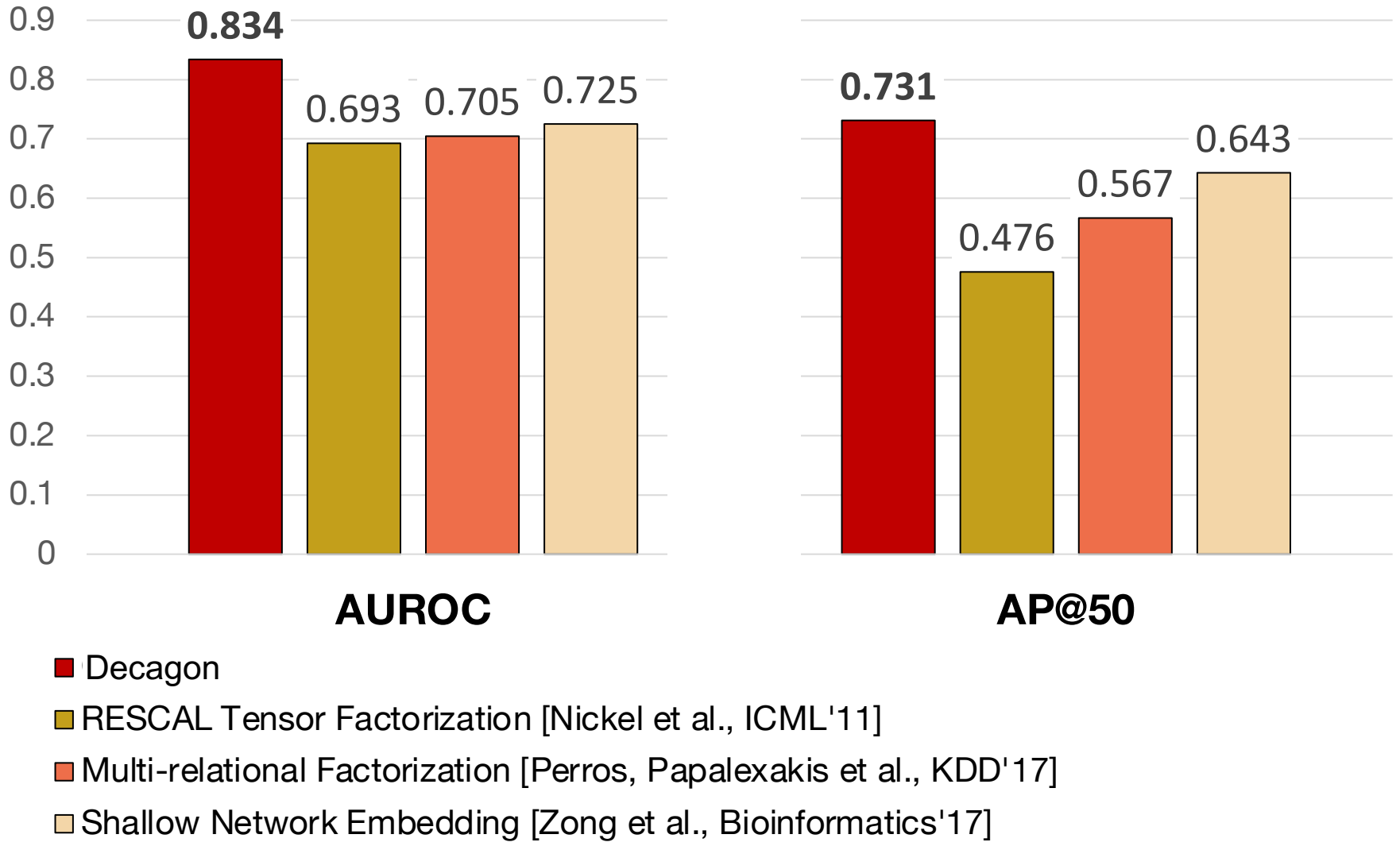
Example: How likely will Simvastatin and Ciprofloxacin, when taken together, break down muscle tissue?

Approach: Graph Neural Network



Map nodes to d-dimensional embeddings such that nodes with similar network neighborhoods are embedded close together

Results: Polypharmacy side effects

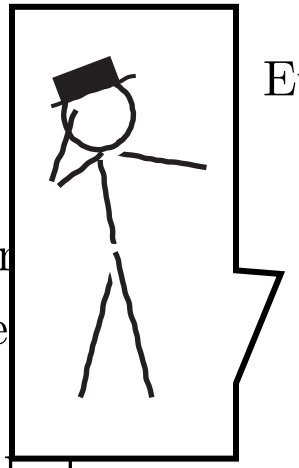


Results: Polypharmacy side effects

Approach:

1) Train deep model on data generated **prior to 2012**
2) How many **predictions** have been **confirmed after 2012**?

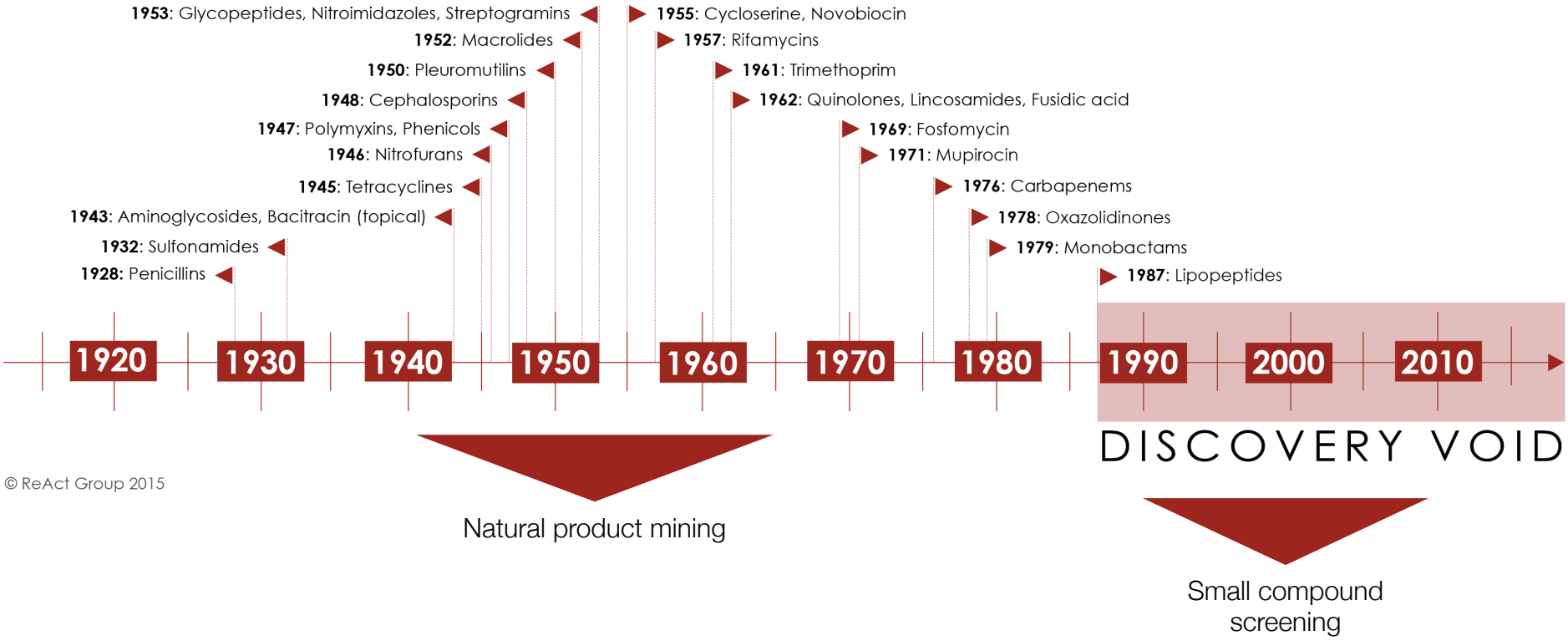
Rank	Drug	Drug	Side effect	Evidence found
1	Pyrimethamine	Aliskiren	Sarcoma	
2	Tigecycline	Bimatoprost	Autonomic r	
3	Telangiectases	Omeprazole	Dacarbazine	
4	Tolcapone	Pyrimethamine	Blood brain	
			eadache	
			ular acidosis	
7	Anag	Azelaic acid	Cerebral thrombosis	
8	Atorvastatin	Amlodipine	Muscle inflammation	
9	Aliskiren	Tioconazole	Breast inflammation	
10	Estradiol	Nadolol	Endometriosis	

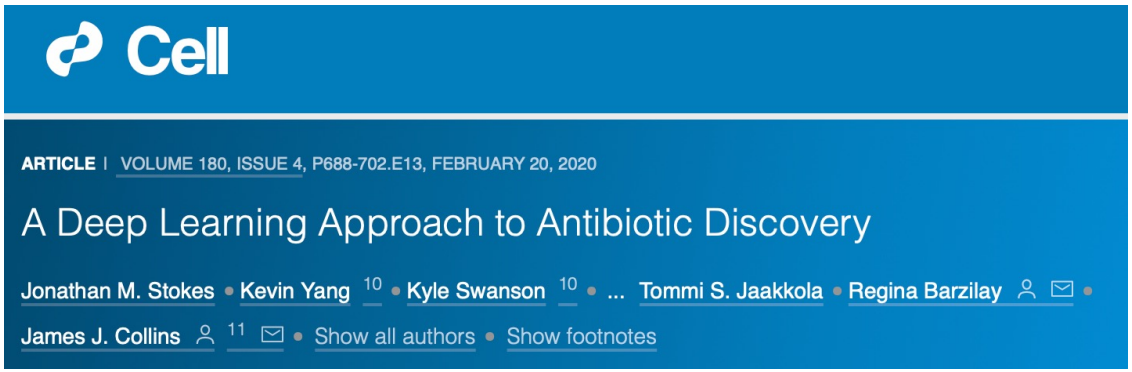


Case Report

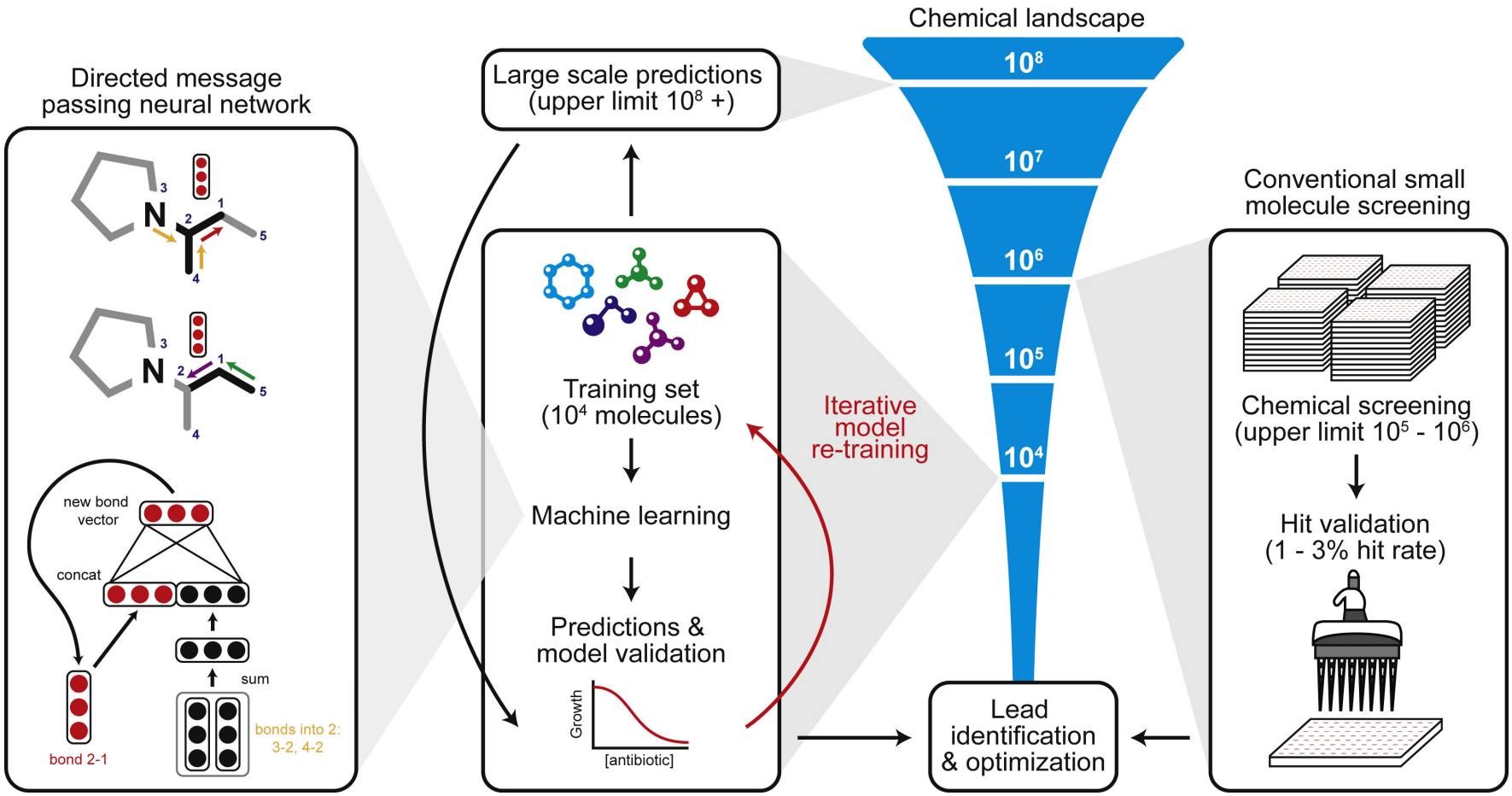
Severe Rhabdomyolysis due to Presumed Drug Interactions between Atorvastatin with Amlodipine and Ticagrelor

Application: Antibiotic discovery





GNNs to learn molecular structure

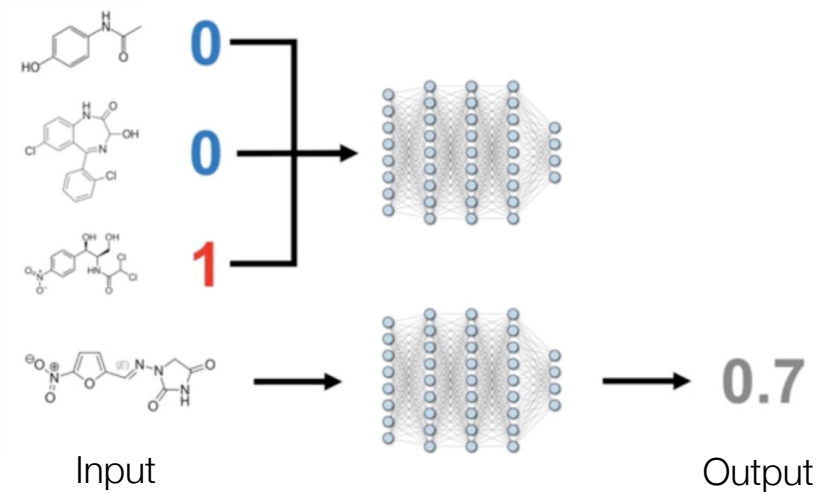


Directed message passing neural network model iteratively (1) learns representations of molecules and (2) optimizes the representations for predicting growth inhibition

A Deep Learning Approach to Antibiotic Discovery, *Cell*, 2020.

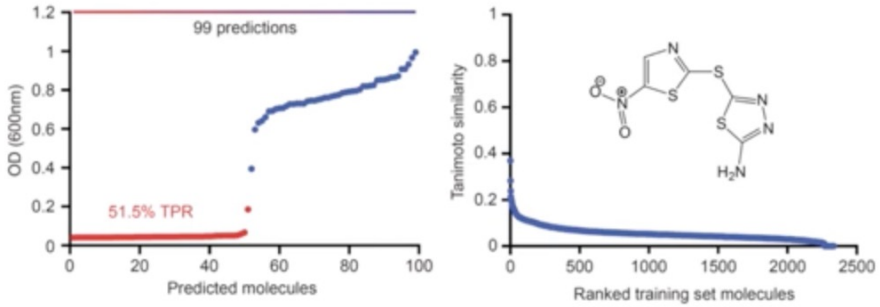
Experimental setup

Training Dataset
(Human Medicines and Natural Products)



Data: 2,335 molecules (human medicines and natural products) screened for growth inhibition

Empirical Validation
(Broad Repurposing Hub)



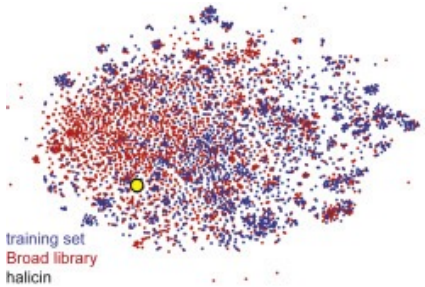
Data: 6,111 molecules (at various stages of investigation for human diseases) in Broad Repurposing Hub

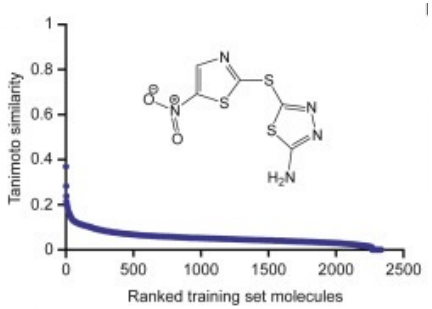
Task: Test top 99 predictions & prioritize based on similarity to known antibiotics or predicted toxicity

Results

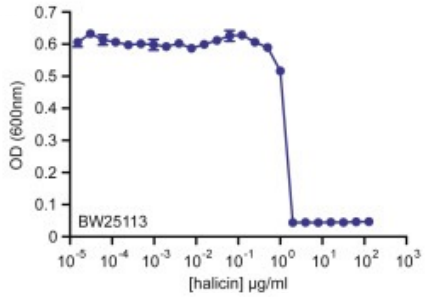
Halicin was developed to be an anti-diabetic drug, but the development was discontinued due to poor results in testing.

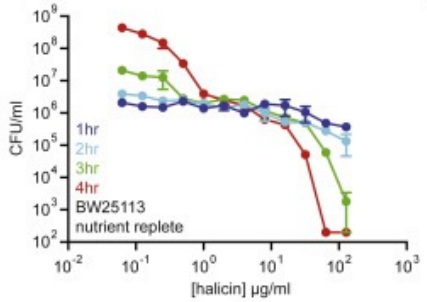
Halicin predicted to be antibacterial



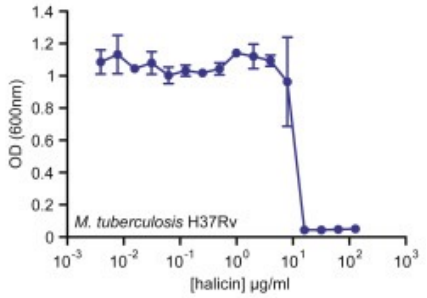


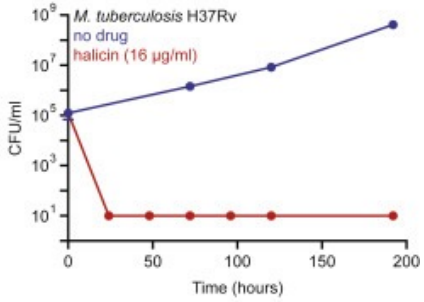
Halicin against *E. coli*





Halicin against *M. tuberculosis*

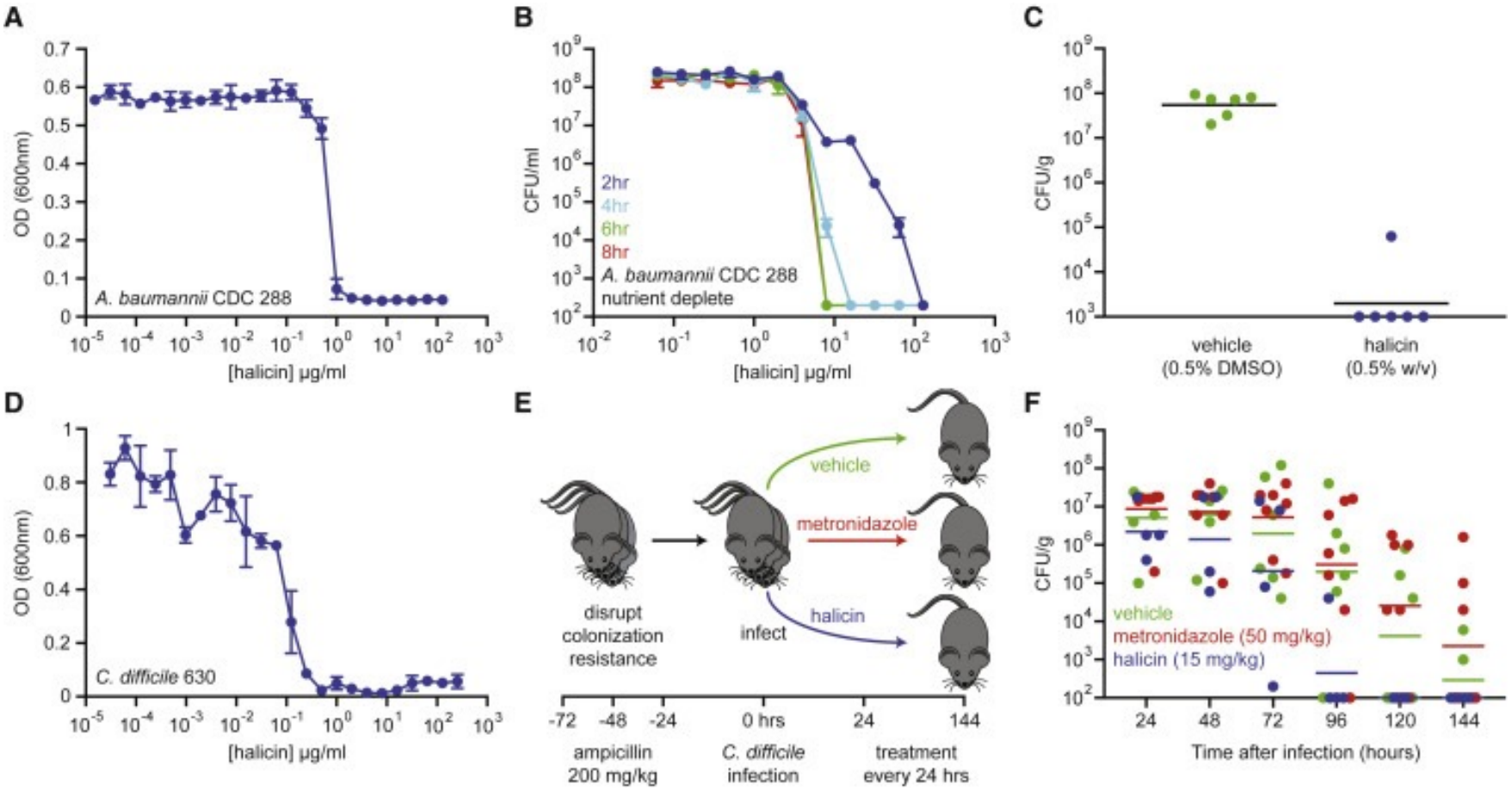




A Deep Learning Approach to Antibiotic Discovery, *Cell*, 2020.

Results

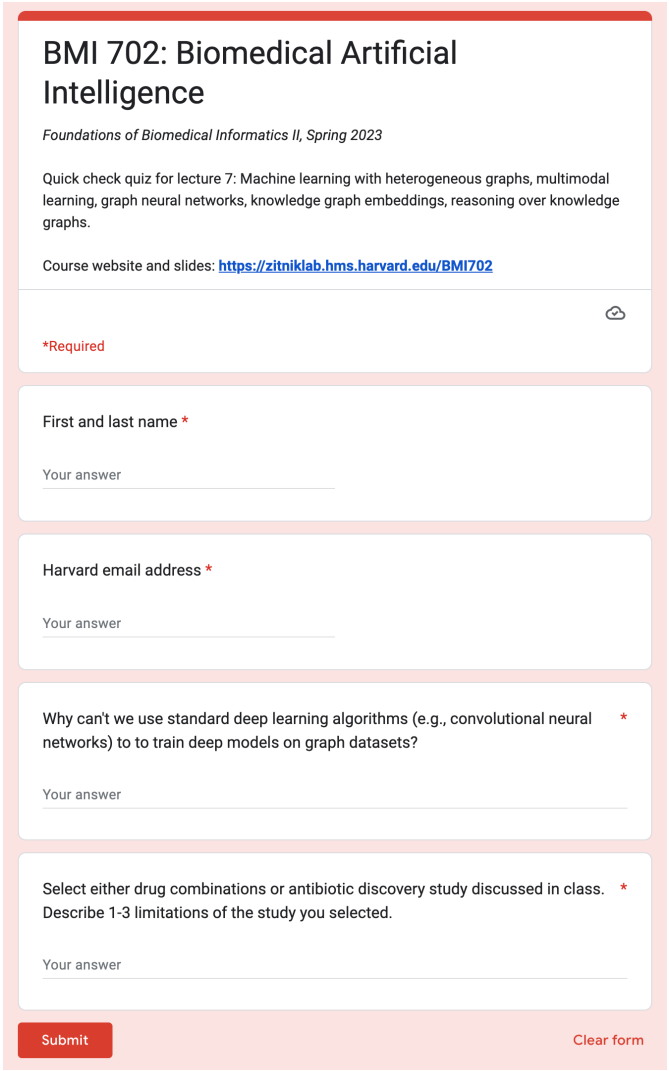
Halicin's efficacy in murine models of infection



Validated against ~6K molecules to identify halicin, a novel candidate antibiotic

Quick Check

https://forms.gle/RYxnxNW5H1TorpU68



BMI 702: Biomedical Artificial Intelligence

Foundations of Biomedical Informatics II, Spring 2023

Quick check quiz for lecture 7: Machine learning with heterogeneous graphs, multimodal learning, graph neural networks, knowledge graph embeddings, reasoning over knowledge graphs.

Course website and slides: https://zitniklab.hms.harvard.edu/BMI702

*Required

First and last name *

Your answer

Harvard email address *

Your answer

Why can't we use standard deep learning algorithms (e.g., convolutional neural networks) to to train deep models on graph datasets? *

Your answer

Select either drug combinations or antibiotic discovery study discussed in class. Describe 1-3 limitations of the study you selected. *

Your answer

Submit

Clear form

Resources

- **Books & survey papers**
  - William Hamilton, Graph Representation Learning (morganclaypool.com/doi/abs/10.2200/S01045ED1V01Y202009AIM046)
  - Li et al., Graph representation learning in biomedicine and healthcare (https://www.nature.com/articles/s41551-022-00942-x)

- **Keynotes & seminars**
  - Michael Bronstein, Geometric Deep Learning: The Erlangen Programme of ML (ICLR 2021 keynote) (youtube.com/watch?v=w6Pw4MOzMuo)
  - Broad Institute Models, Inference & Algorithms: Actionable machine learning for drug discovery; Primer on graph representation learning (youtube.com/watch?v=9YpTYdru0Rg)
  - Stanford University (CS224W Lecture): Graph neural networks in computational biology (youtube.com/watch?v=_hy9AgZXhbQ)
  - AI Cures Drug Discovery Conference (youtube.com/watch?v=wNXSkISMTw8)

- **Conferences & summer schools**
  - London Geometry and Machine Learning Summer School (logml.ai)
  - Learning on Graphs Conference (logconference.github.io)

Resources

- **Software & packages**
  - PyTorch Geometric
  - NetworkX
- **Tutorials & code bases**
  - Precision Medicine with Graph Representation Learning (https://zitniklab.hms.harvard.edu/biomedgraphml)
  - 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)

Outline for today's lecture

- **Methods:**
  - Shallow graph embeddings
  - Deep graph neural networks
  - Knowledge graph learning
- **Part 1: Applications in biomedical AI**
  - Polypharmacy side effects
  - Antibiotic discovery
- **Part 2: Applications in clinical AI [Michelle M. Li]**
  - Rare genetic disease diagnosis
  - Patient disease state prediction
  - Medication recommendation