AIM 2: Artificial Intelligence in Medicine II

Harvard - BMIF 203 and BMI 702, Spring 2025

Lecture 8: Foundations of network biology and medicine, Foundations of graph AI, Semi-supervised learning and label diffusion, Graph representation learning, Introduction to graph neural networks (GNNs), Neural message-passing models, Applications in gene function prediction, medical diagnosis, drug combination modeling, and antibiotic discovery

HARVARD MEDICAL SCHOOL

Kempner INSTITUTE For the Study of Natural & Artificial Intelligence at Harvard University

BROAD INSTITUTE

Marinka Zitnik
marinka@hms.harvard.edu

Outline for today's class

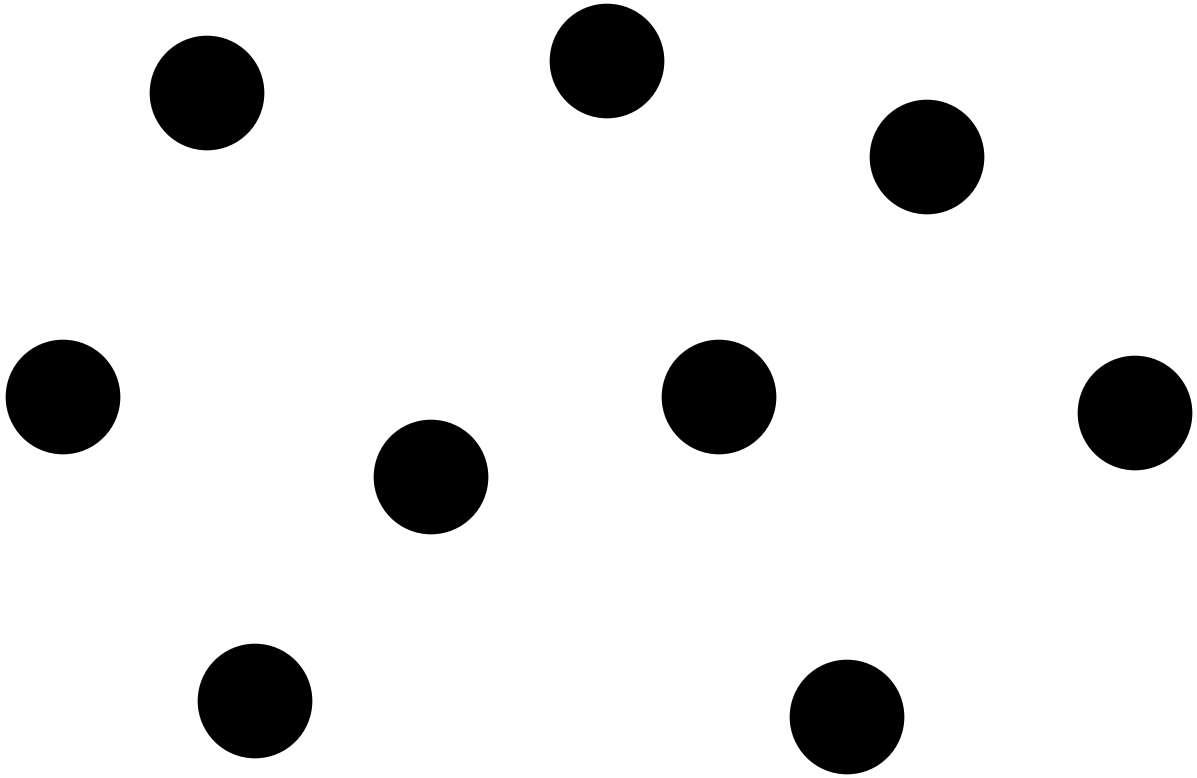
- **Foundations of network biology and medicine**
- **Foundations of graph AI**
  - Node classification, link prediction, graph classification
  - Semi-supervised learning and label diffusion
- **Graph representation learning**
  - Shallow graph embeddings
  - Introduction to graph neural networks (GNNs)
  - Neural message-passing models
- **Applications**
  - **Gene function prediction:** What does my gene do?
  - **Medical diagnosis:** Patients-like-me retrieval and diagnosis
  - **Drug combination modeling:** polypharmacy
  - **Antibiotic discovery:** Finding new candidate antibiotics

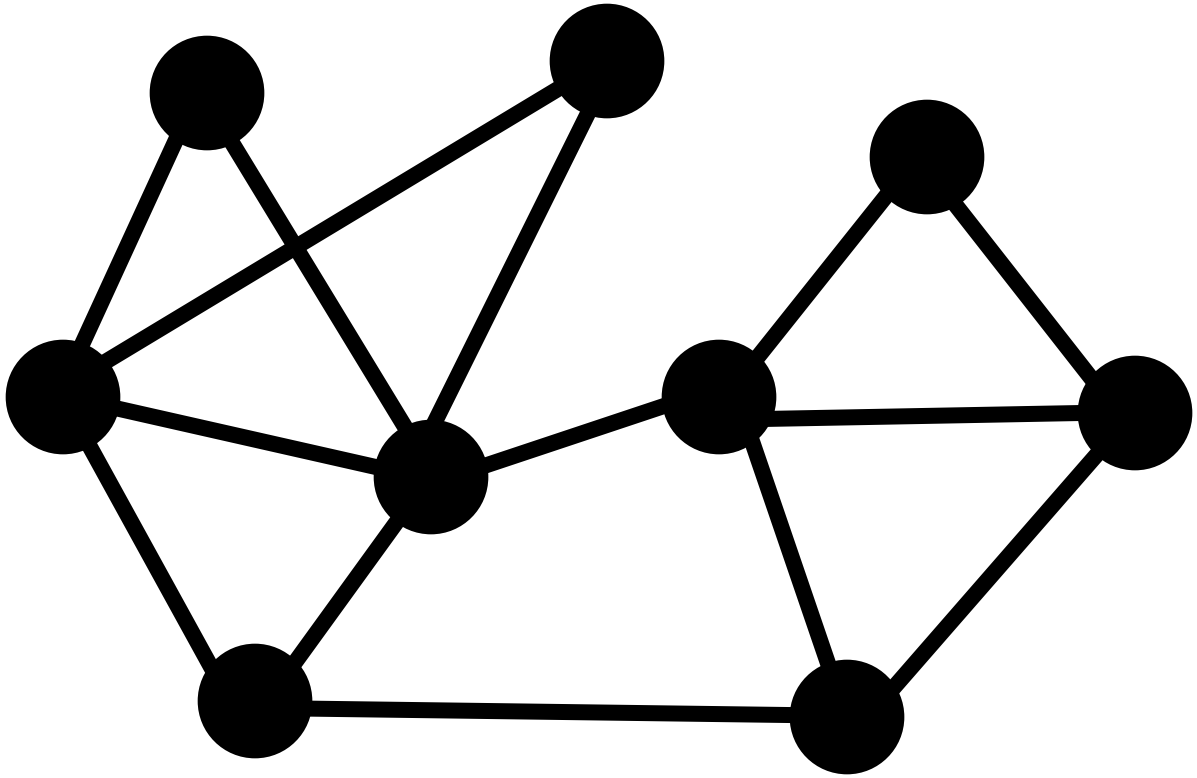
Foundations of network biology and medicine

What are networks/graphs?
Predictive modeling using graphs

Why networks?

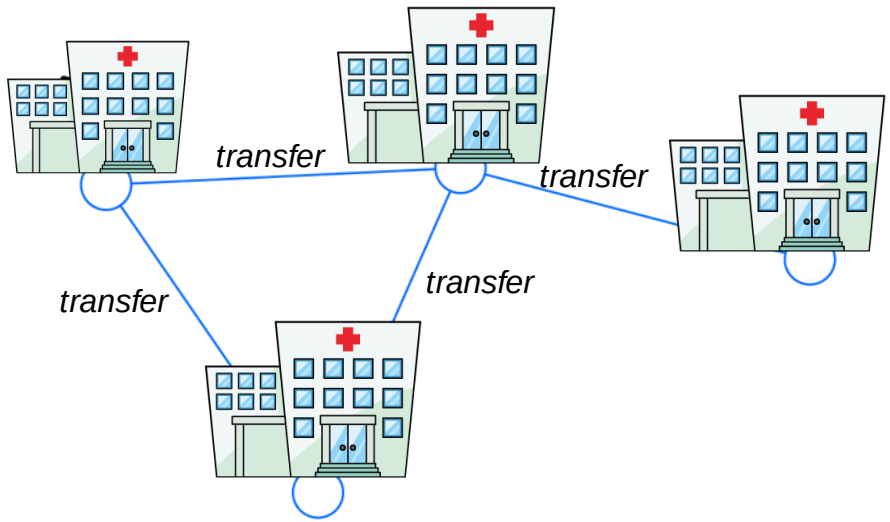
Networks are a general language for describing and modeling complex systems

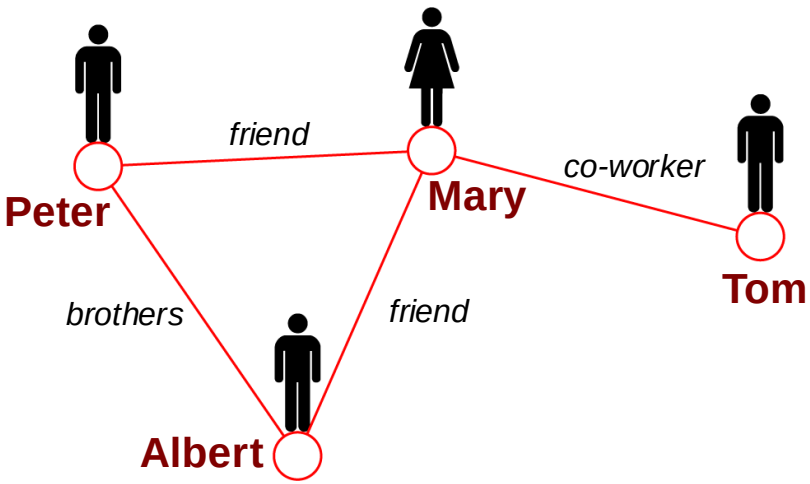


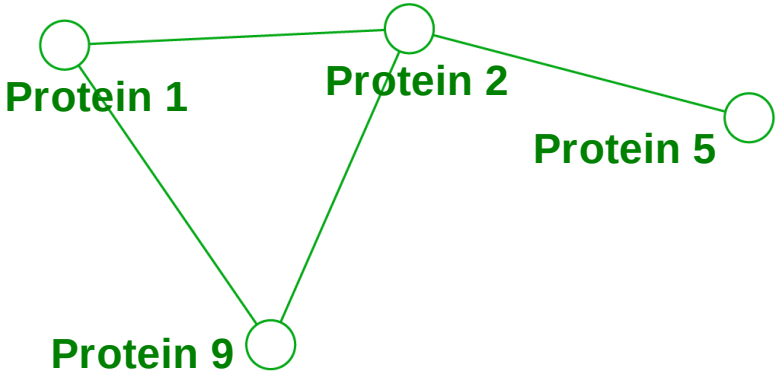


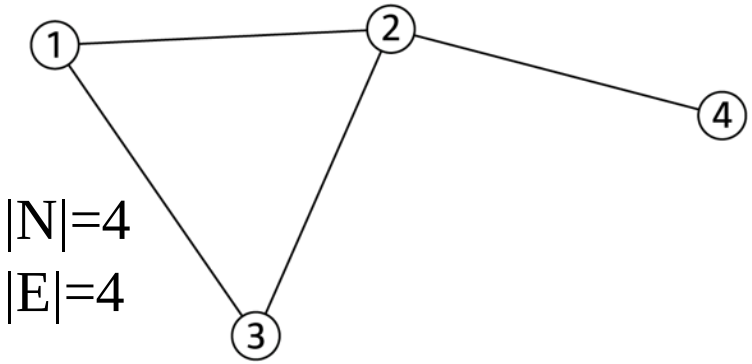
Network!

General Mathematical Language



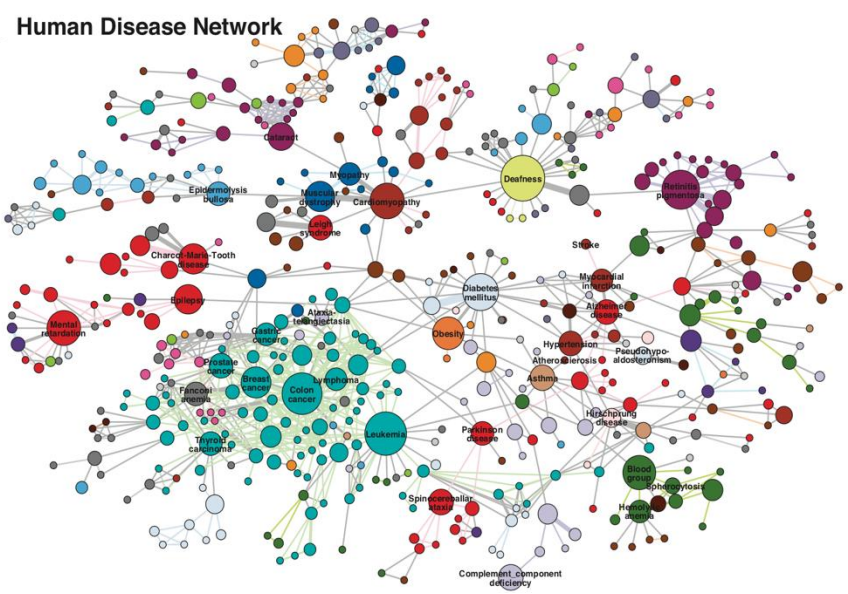






Why Networks? Why Now?

- **Question:** How are diseases and disease genes related to each other?
- **Findings:** Disease genes likely to interact and have similar expression



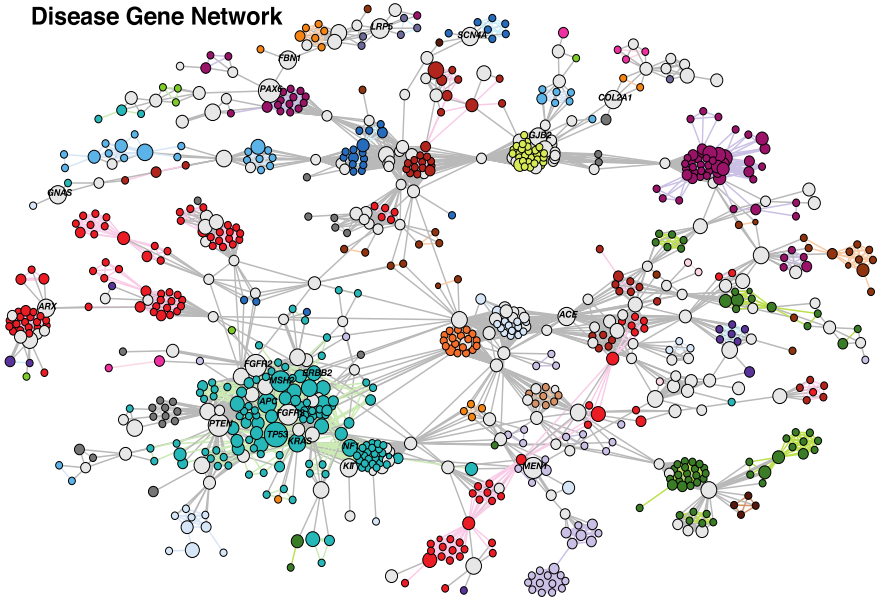


Image from: Goh et al. 2007. The human disease network. *PNAS*.

Why Networks? Why Now?

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

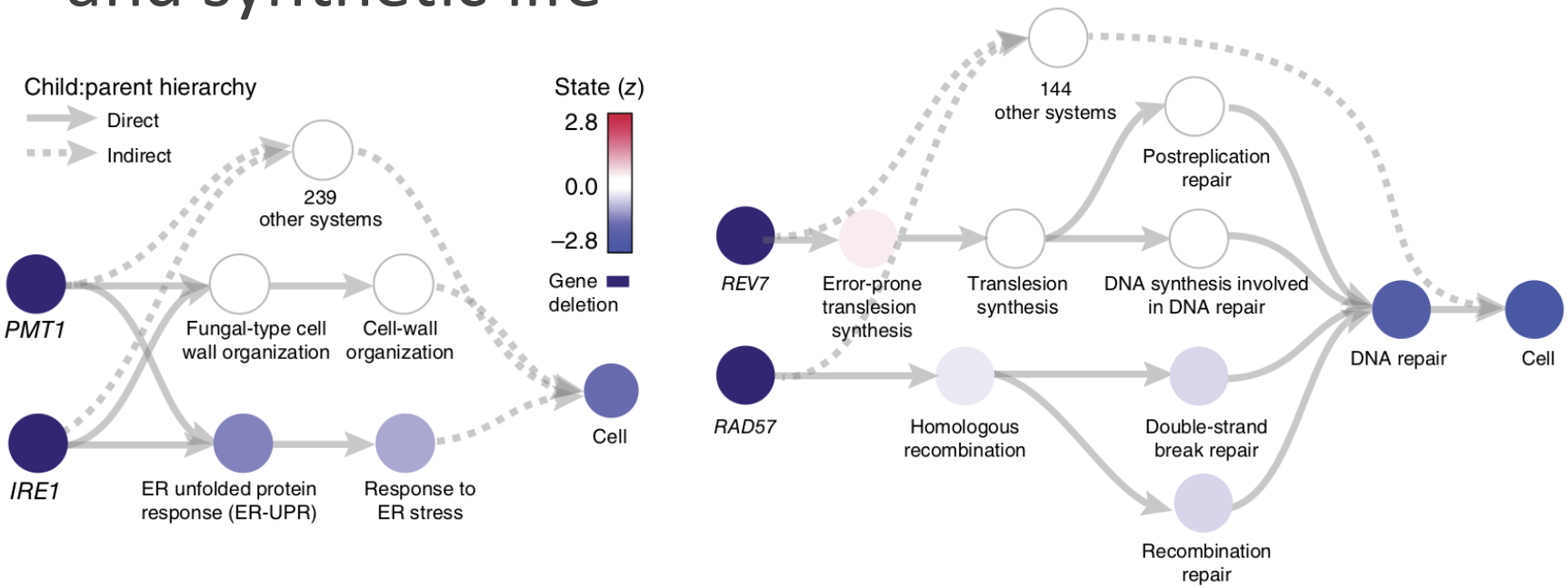


Image from: Ma et al. 2018. Using deep learning to model the hierarchical structure and function of a cell. *Nature Methods*.

Why Networks? Why Now?

- **Question:** How to model cancer heterogeneity?
- **Findings:** New cancer subtypes with distinct patient survival

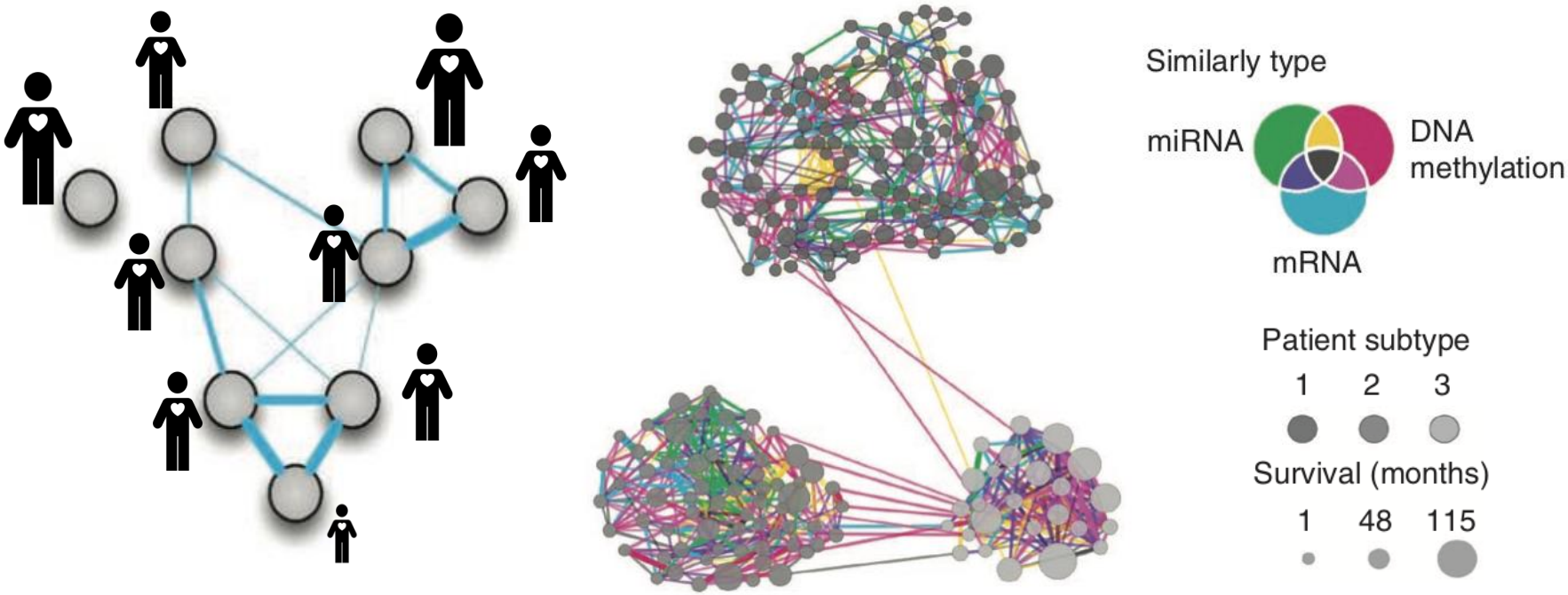
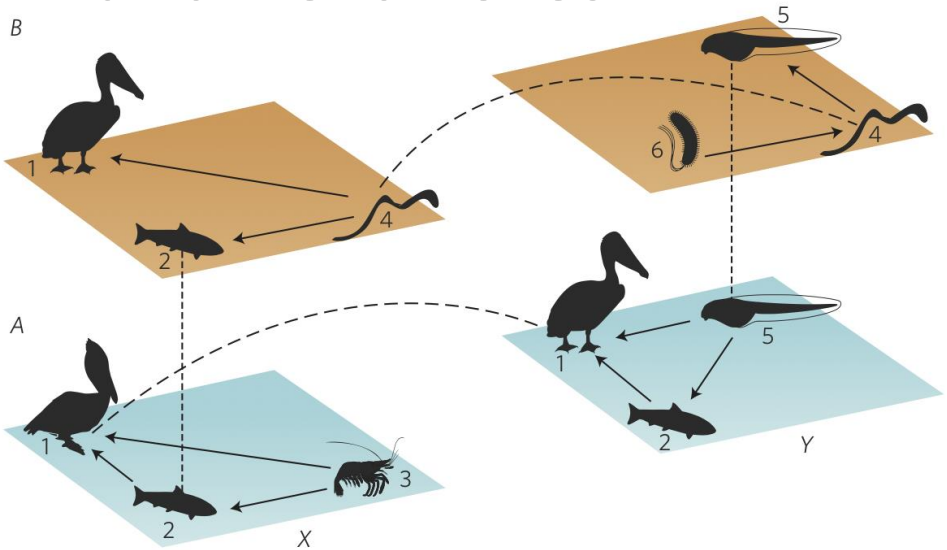


Image from: Wang et al. 2014. Similarity network fusion for aggregating data types on a genomic scale. *Nature Methods*.

Why Networks? Why Now?

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



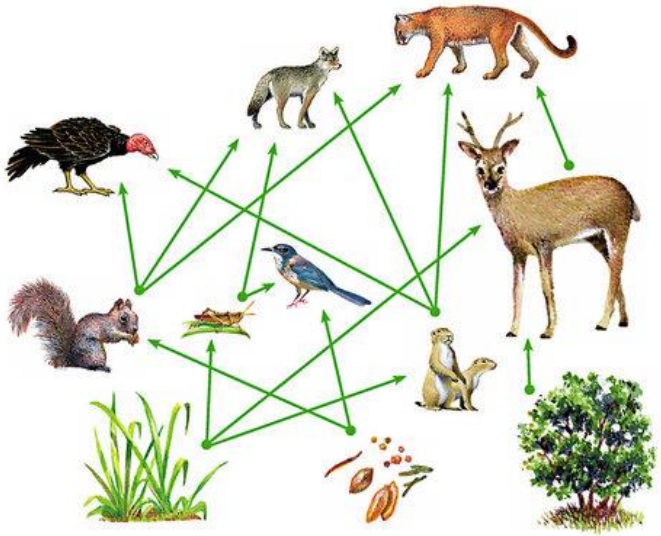
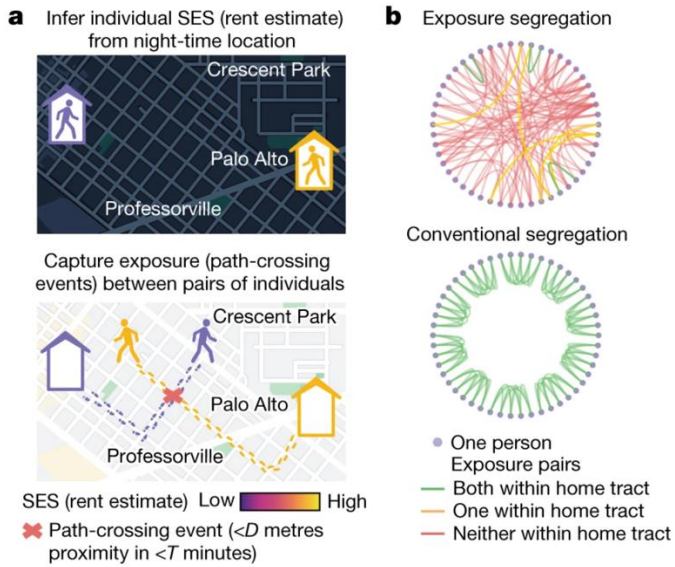


Image from: Pilosof et al. 2017. The multilayer nature of ecological networks. *Nature Ecology and Evolution*.

Why Networks? Why Now?

- **Question:** Do large, dense, and cosmopolitan areas support socioeconomic mixing and exposure among diverse individuals?
- **Findings:** Contrary to expectations, residents of large cosmopolitan areas have less exposure to a socioeconomically diverse range of individuals



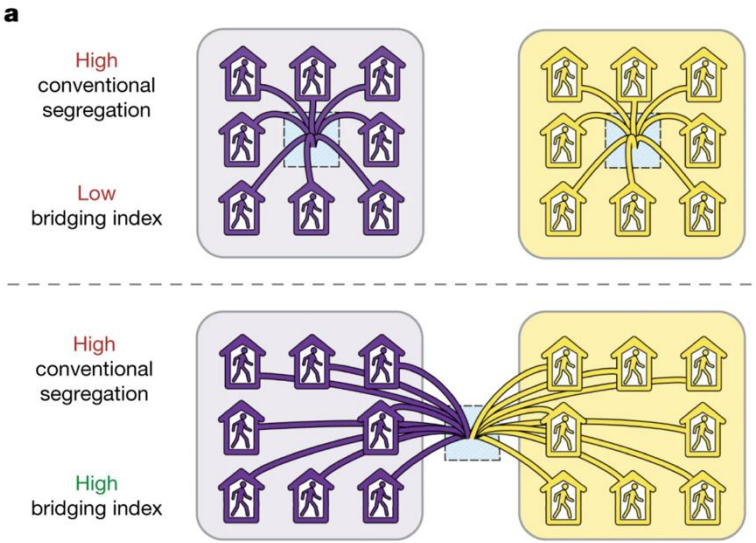
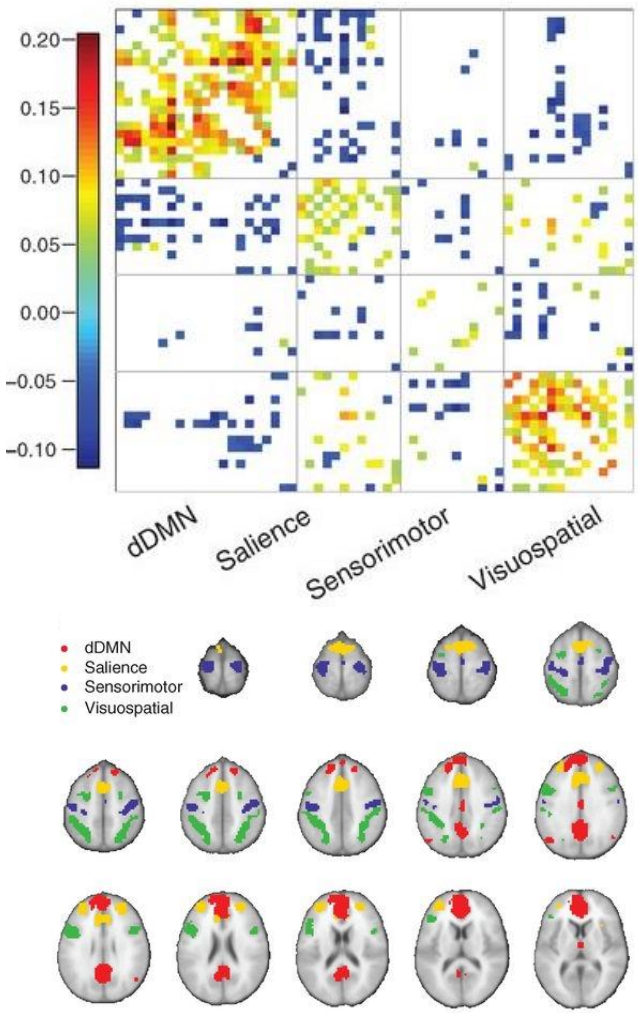


Image from: Nilforoshan et al. 2023. Human mobility networks reveal increased segregation in large cities. *Nature*.

Why Networks? Why Now?



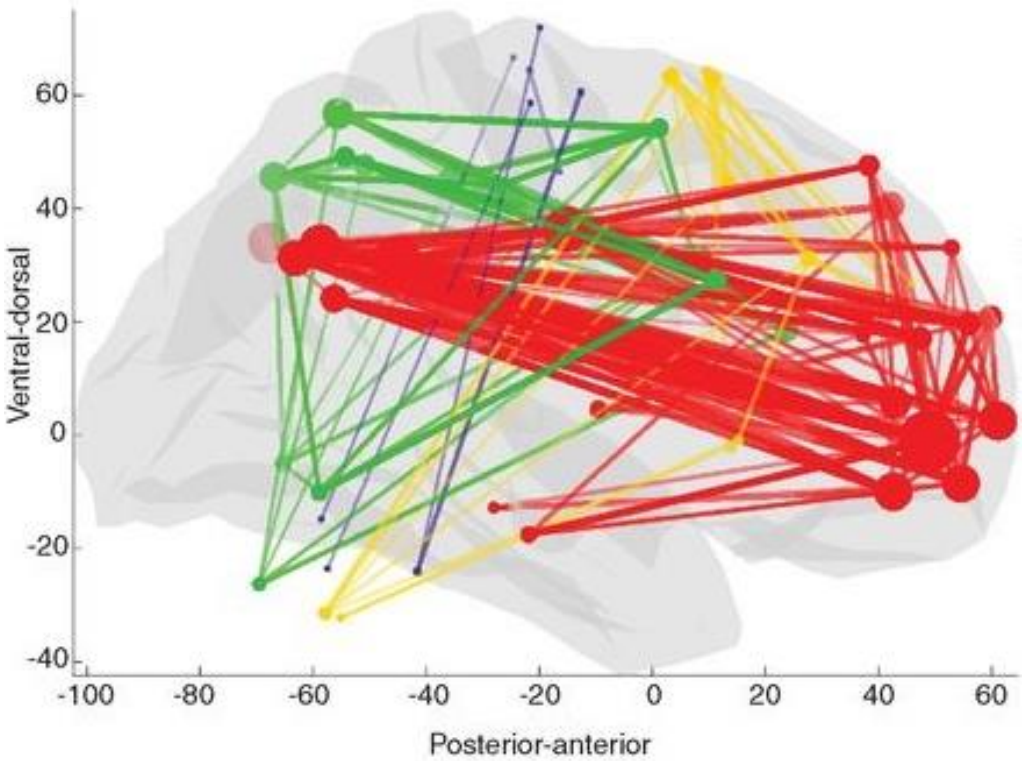
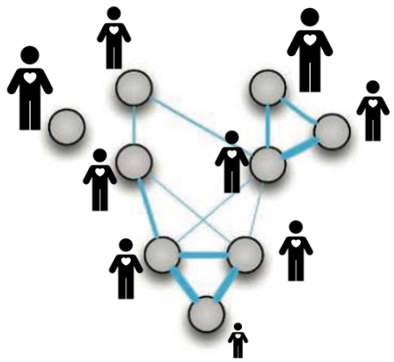
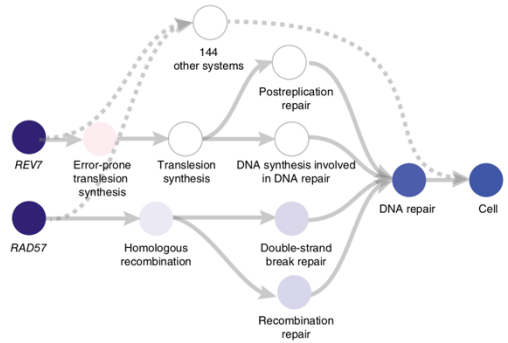


Image from: Richiardi et al. 2015. Correlated gene expression supports synchronous activity in brain networks. *Science*.

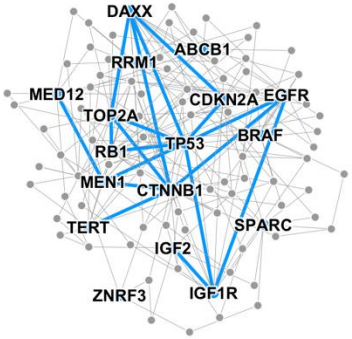
Many Data are Networks



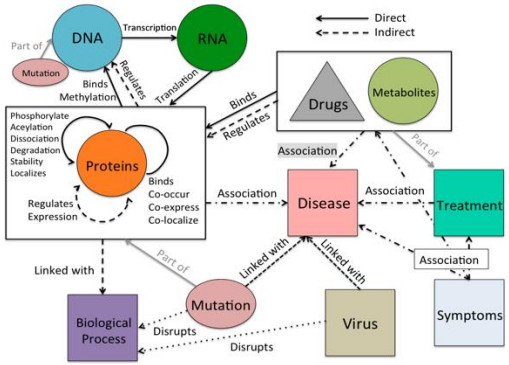
Patient networks



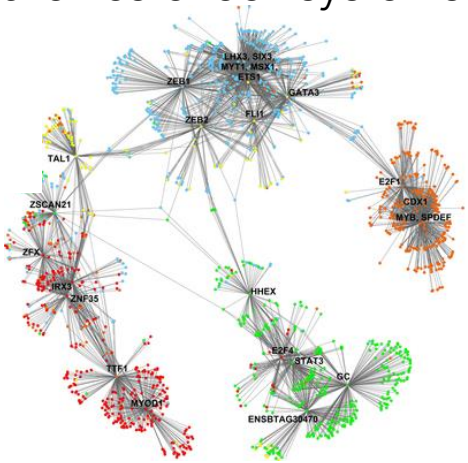
Hierarchies of cell systems



Disease pathways



Biomedical knowledge graphs



Gene interaction networks

Cell-cell similarity networks

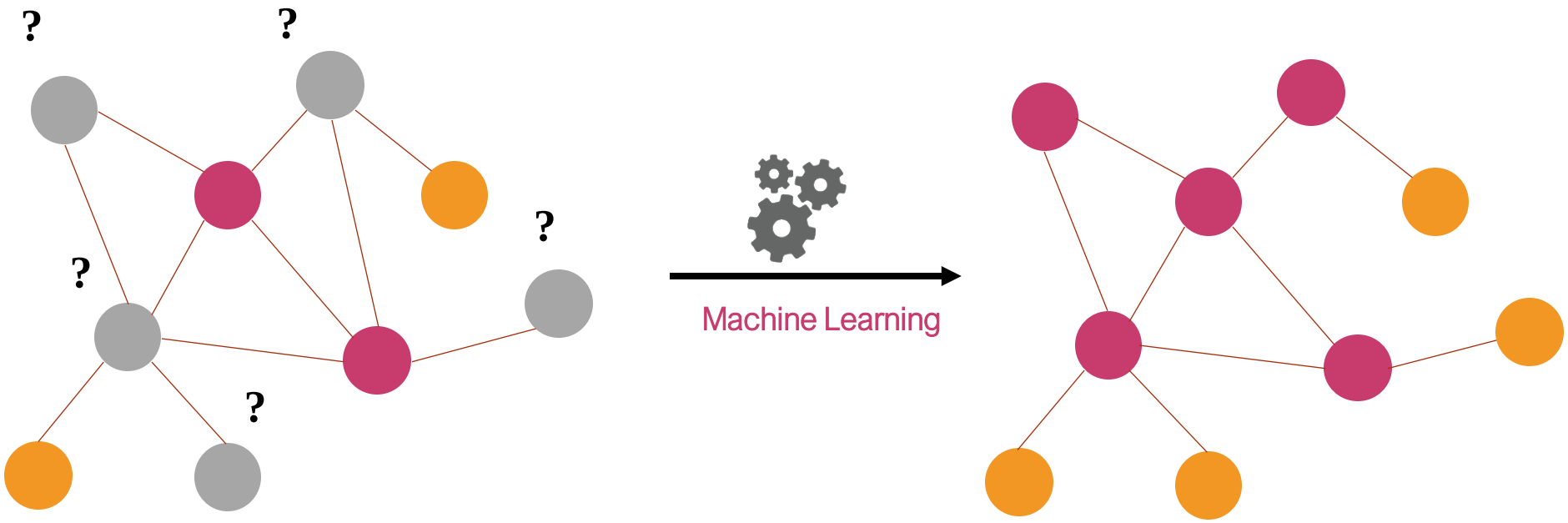
Evolution of Resilience in Protein Interactomes Across the Tree of Life, *PNAS,* 2019; MARS: Discovering Novel Cell Types across Heterogeneous Single-Cell Experiments, *Nat Methods*, 2020; Leveraging the Cell Ontology to Classify Unseen Cell Types, *Nat Commun*, 2021; Identification of Disease Treatment Mechanisms through the Multiscale Interactome, *Nat Commun,* 2021; Network Medicine Framework for Identifying Drug Repurposing Opportunities for COVID-19, *PNAS,* 2021; Population-Scale Patient Safety Data Reveal Inequalities in Adverse Events Before and During COVID-19 Pandemic, *Nat Comput Science*, 2021

Predictive and Generative Modeling

- **Predict a type of a given node**
  - Node classification
- **Predict whether two nodes are linked**
  - Link prediction
- **Identify densely linked clusters of nodes**
  - Community detection, module detection
- **How similar are two nodes/networks**
  - Network similarity
- **Design graphs with desirable properties**
  - Generative modeling and molecular design

    This topic will be covered in M6: Generative AI

Node Classification



Node Classification: Example

Classifying the function of proteins in the interactome!

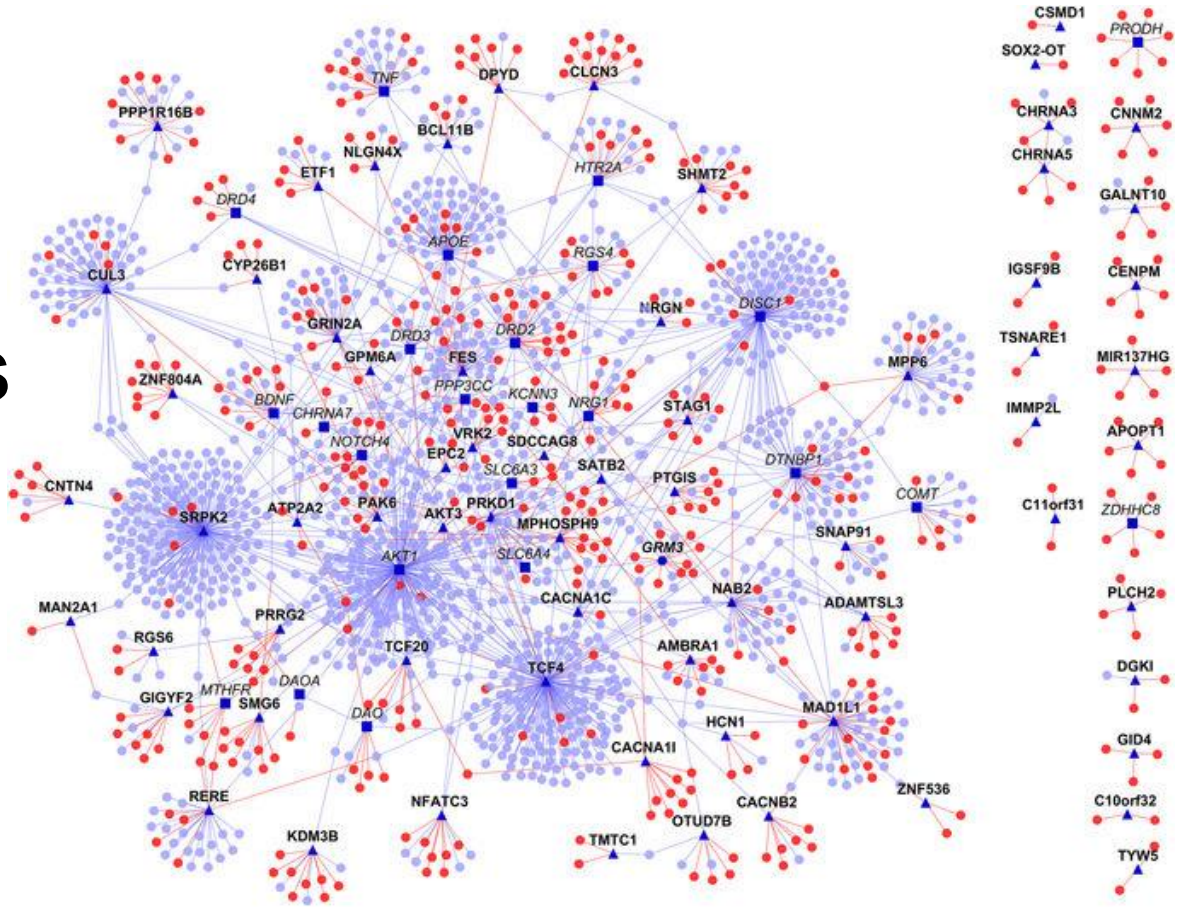
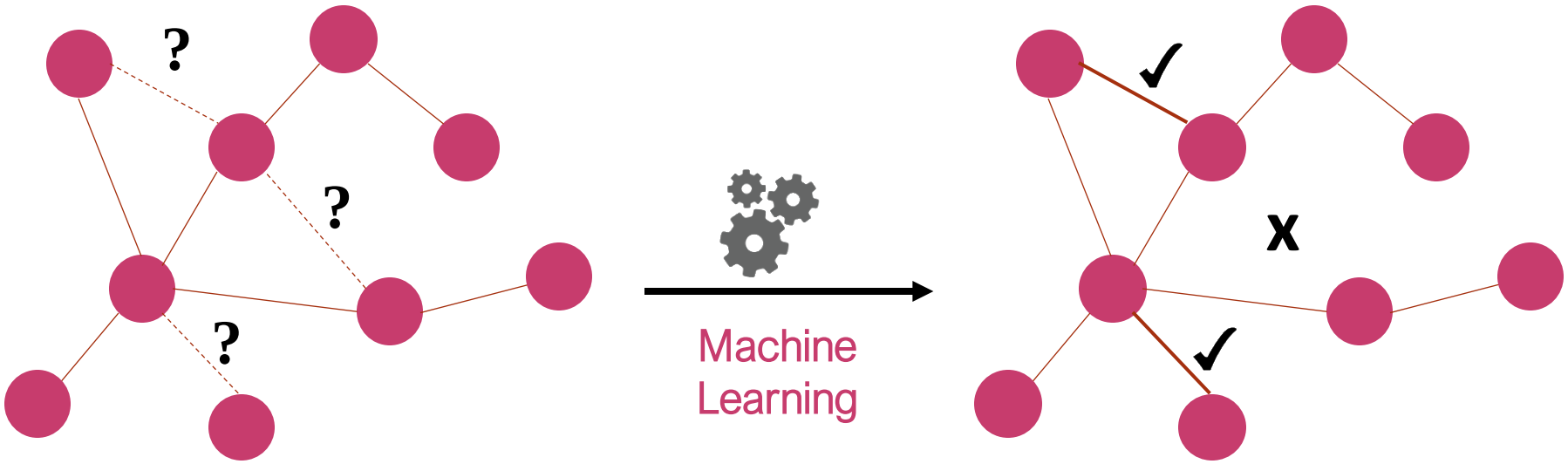


Image from: Ganapathiraju et al. 2016. Schizophrenia interactome with 504 novel protein–protein interactions. *Nature*.

Link Prediction



Link Prediction: Example

Predicting which diseases a new molecule might treat!

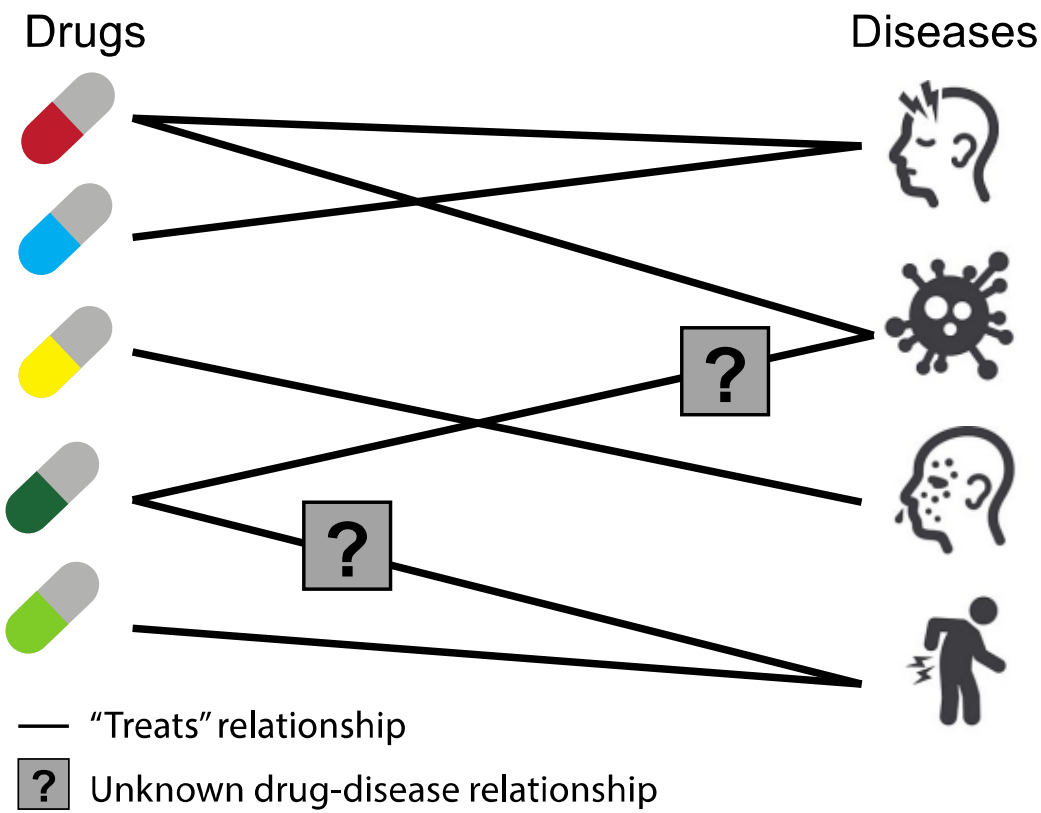
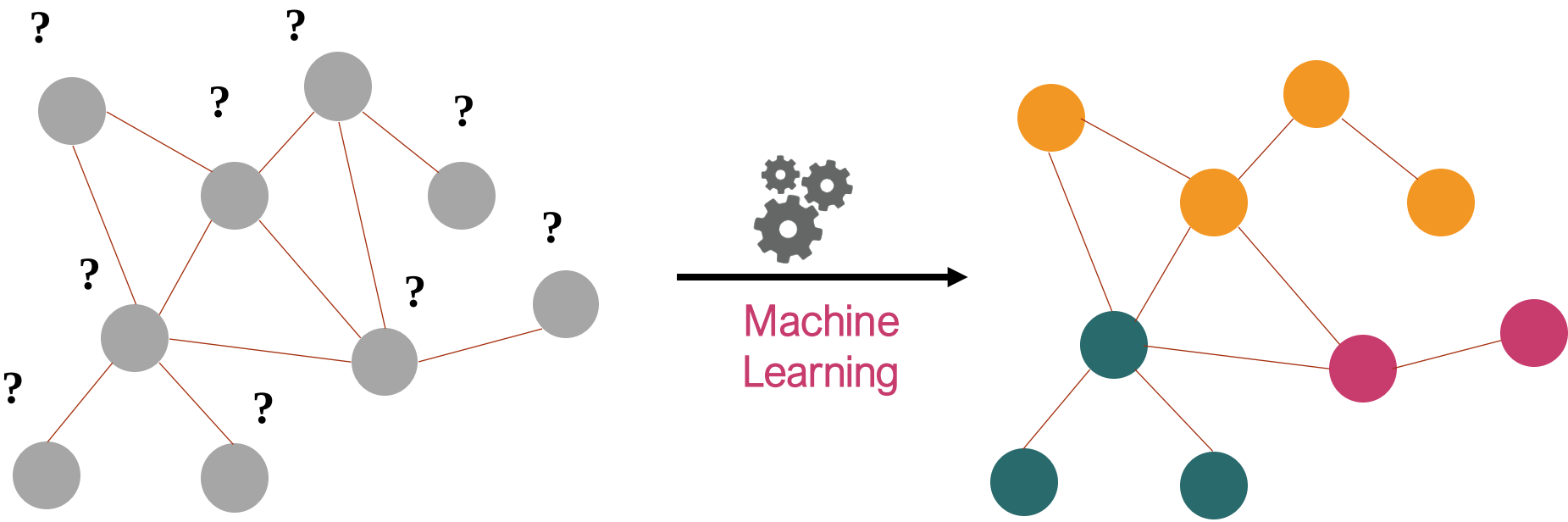


Image from: Zitnik et al. 2020. Network-based discovery of drug indications.

Community Detection



Community Detection: Example

Identifying disease proteins in the interactome!

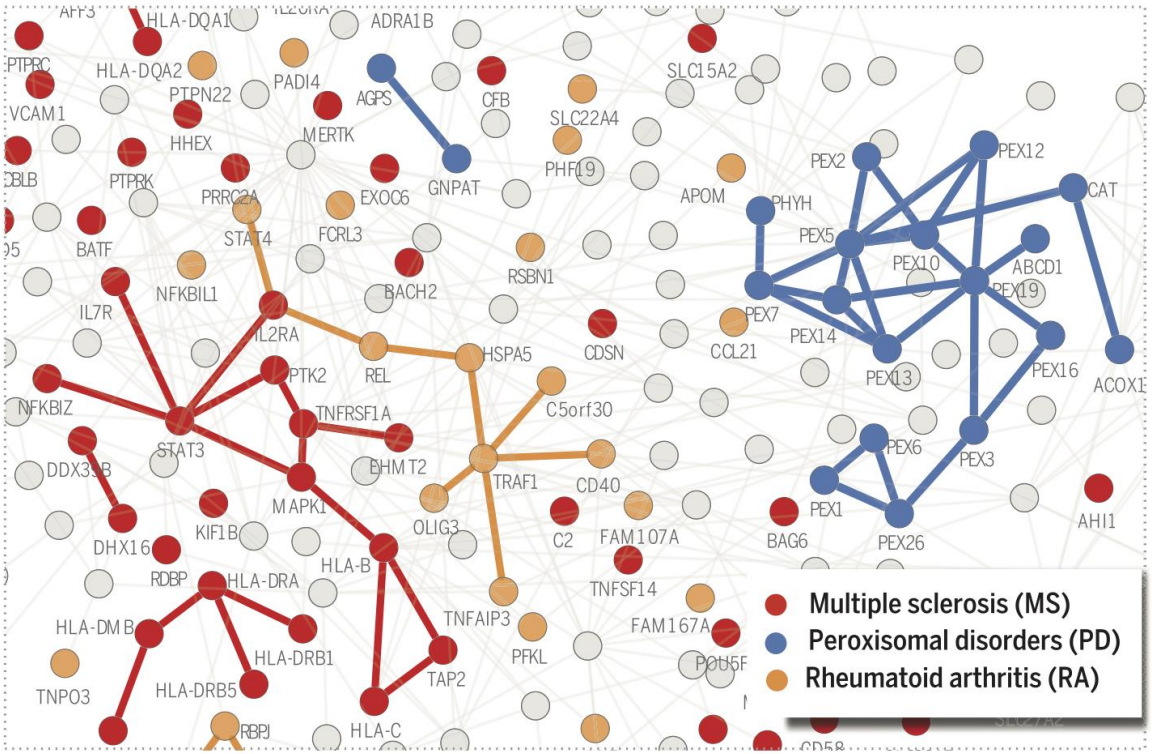
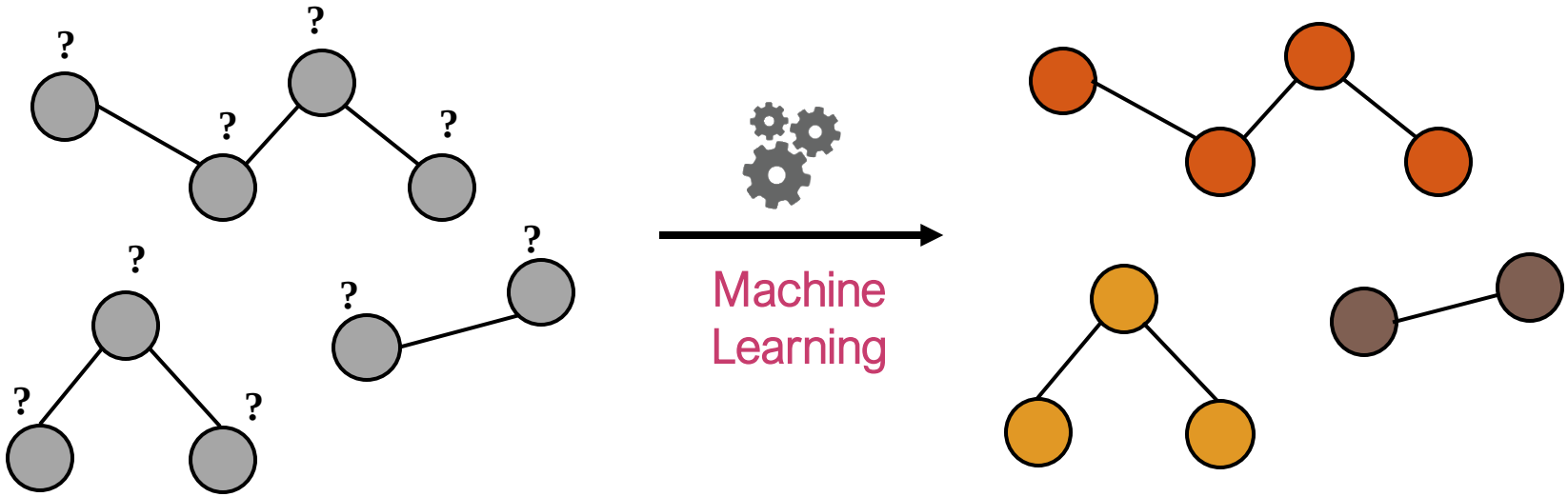


Image from: Menche et al. 2015. Uncovering disease-disease relationships through the incomplete interactome. *Science*.

Graph Classification



Graph Classification: Example

Designing new small molecule compounds to treat a disease!

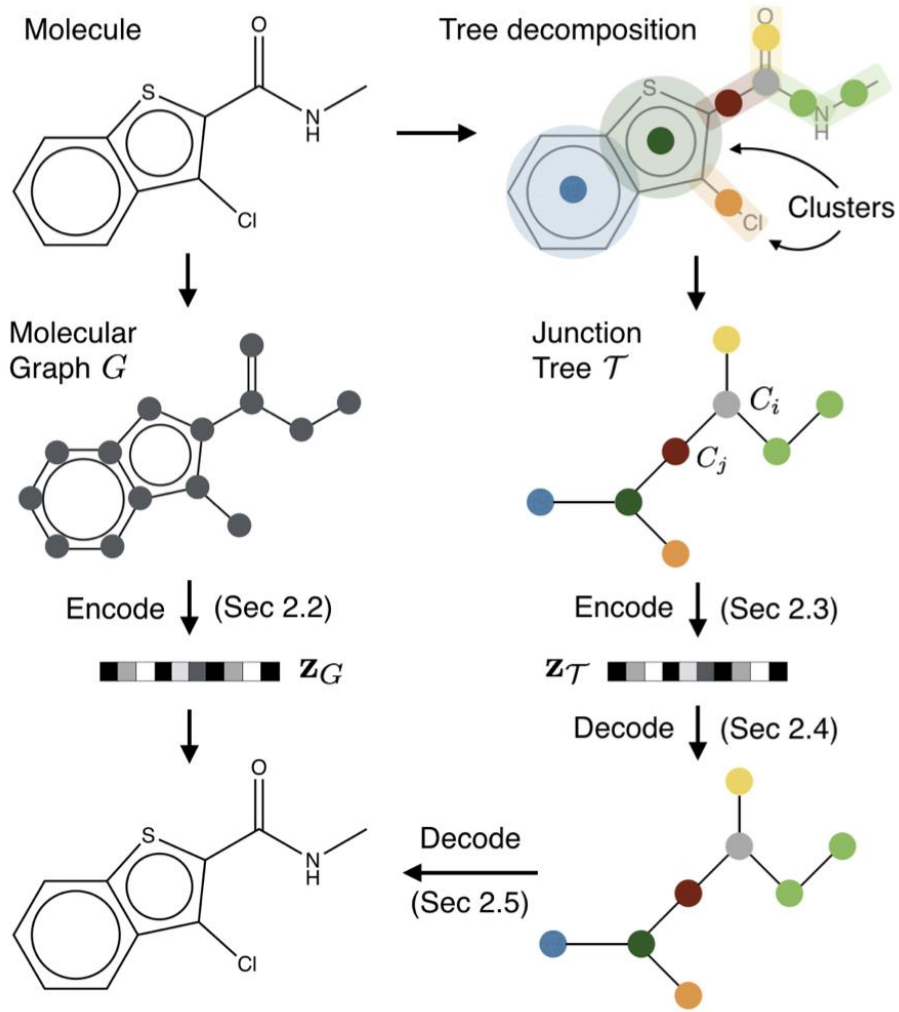
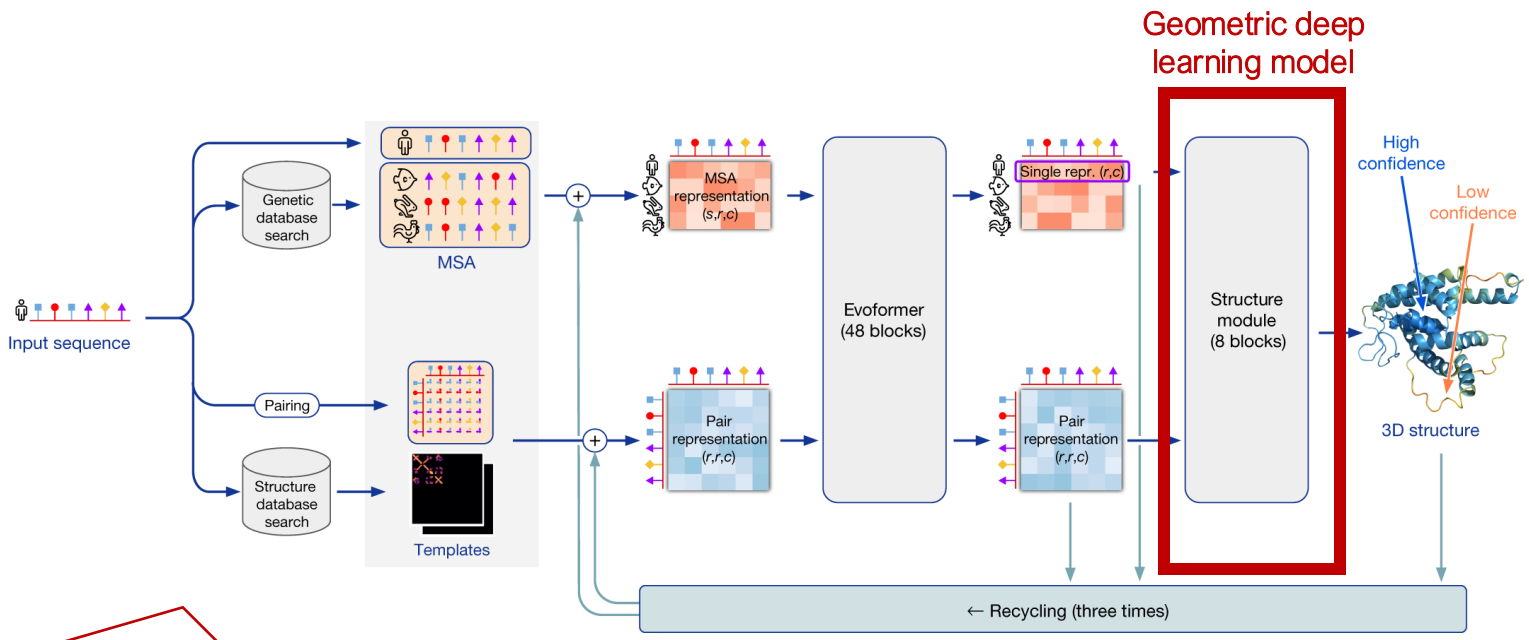


Image from: Jin et al. 2018. Junction Tree Variational Autoencoder for Molecular Graph Generation. *ICML.*

Generative Modeling and Design

Geometric deep learning underlies several breakthroughs, including AlphaFold for protein structure prediction



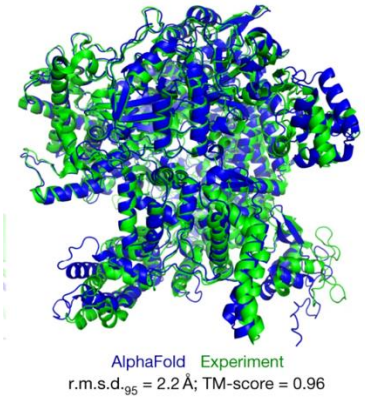
Geometric deep learning is receiving increasing interest in biology, chemistry, and medical sciences as a new tool for molecular design and optimization

Jumper et al., Highly accurate protein structure prediction with AlphaFold, Nature 2021

AlphaFold Network

- **What drives accurate protein structure prediction?**
  - Novel neural architecture based on the **evolutionary**, **physical** and **geometric constraints** of protein structures
- Input:
  - Primary AA sequence of a given protein
  - Aligned sequences of homologues

- Output:
  - Predicted 3D coordinates of all heavy atoms in a protein



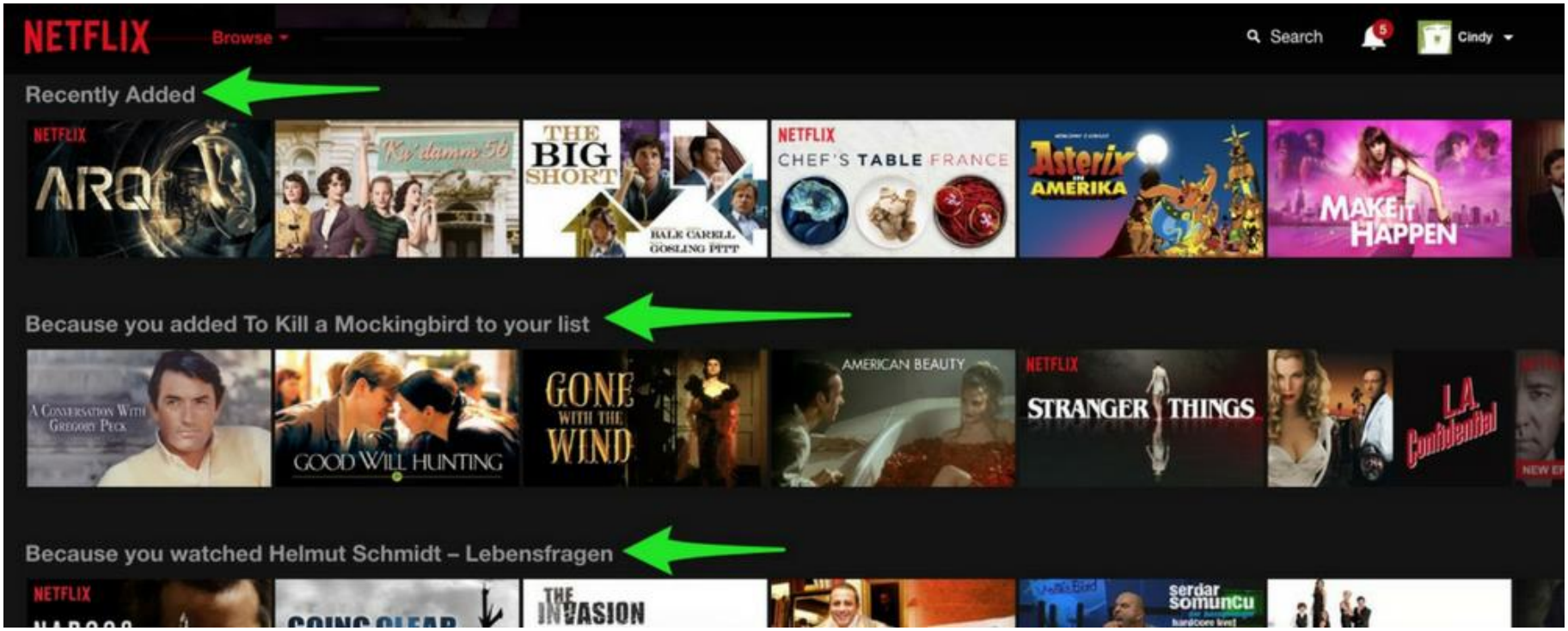
Jumper et al., Highly accurate protein structure prediction with AlphaFold, Nature 2021

Genes-like-me

What does my gene do? Give me more genes like these

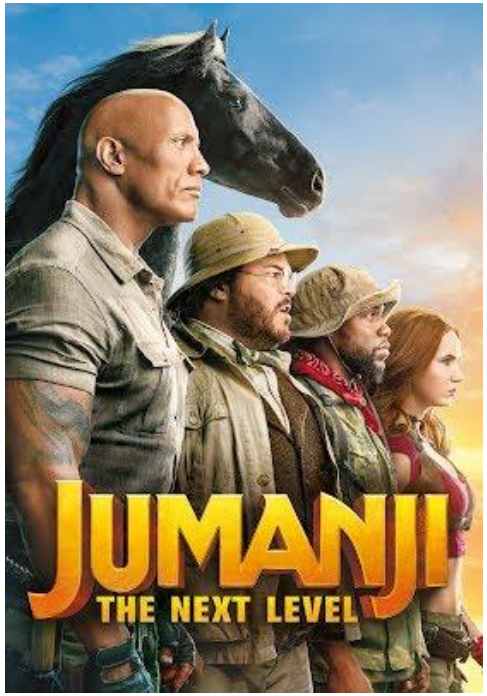
Recommender Systems

Consider user ***x:*** Find set ***S*** of other users whose ratings are **"similar"** to ***x***'s ratings; Estimate ***x***'s preference based on ratings in ***S***

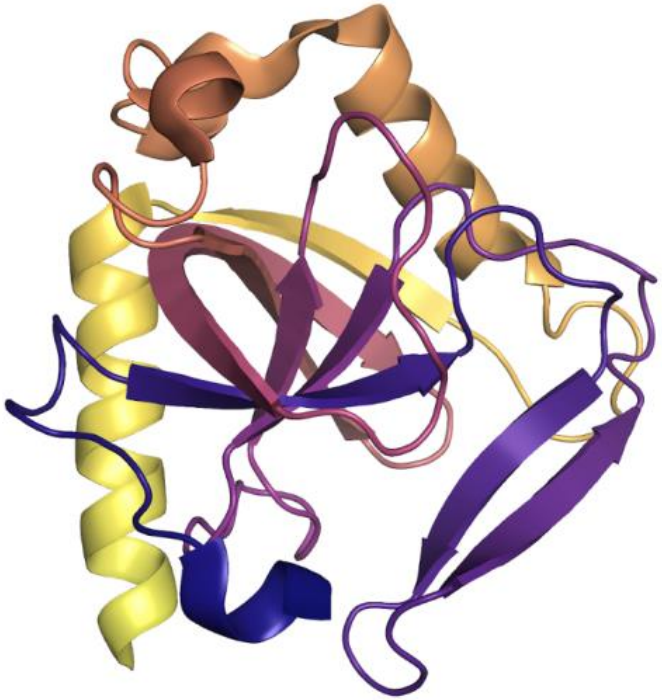


Recommender Systems in Biology

"Give me more <u>movies</u> like this one"

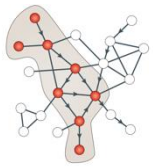


"Give me more <u>proteins</u> like this one"

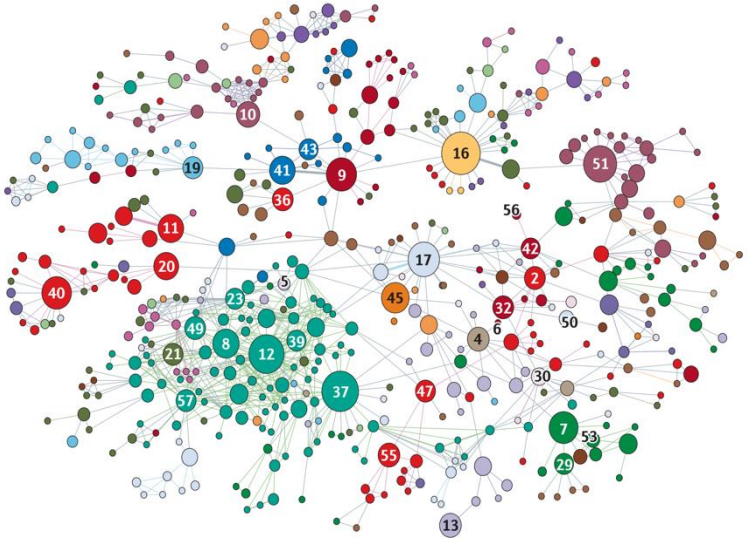


Biological Rationales

- **Local hypothesis:** Proteins involved in the same disease have an increased tendency to interact with each other



- **Disease module hypothesis:** Cellular components associated with disease tend to cluster in the same network neighborhood



1. Aldosteronism
2. Alzheimer's disease
3. Anaemia, congenital deserythropoietic
4. Asthma
5. Ataxia-telangiectasia
6. Atherosclerosis
7. Blood group
8. Breast cancer
9. Cardiomyopathy
10. Cataract
11. Charcot–Marie–Tooth disease
12. Colon cancer
13. Complement component deficiency
14. Coronary artery disease
15. Coronary spasm
16. Deafness
17. Diabetes mellitus
18. Enolase-β deficiency
19. Epidermolysis bullosa
20. Epilepsy
21. Fanconi's anaemia
22. Fatty liver
23. Gastric cancer
24. Gilbert's syndrome
25. Glaucoma 1A
26. Goitre congenital
27. HARP syndrome
28. HELLP syndrome
29. Haemolytic anaemia
30. Hirschprung disease
31. Hyperbilirubinaemia
32. Hypertension
33. Hypertension diastolic
34. Hyperthyroidism
35. Hypoaldosteronism
36. Leigh syndrome
37. Leukaemia
38. Low renin hypertension
39. Lymphoma
40. Mental retardation
41. Muscular dystrophy
42. Myocardial infarction
43. Myopathy
44. Nucleoside phosphorylase deficiency
45. Obesity
46. Paraganglioma
47. Parkinson's disease
48. Pheochromocytoma
49. Prostate cancer
50. Pseudohypoaldosteronism
51. Retinitis pigmentosa
52. Schizoaffective disorder
53. Spherocytosis
54. Spina bifida
55. Spinocerebellar ataxia
56. Stroke
57. Thyroid carcinoma
58. Total iodide organification defect
59. Trifunctional protein deficiency
60. Unipolar depression

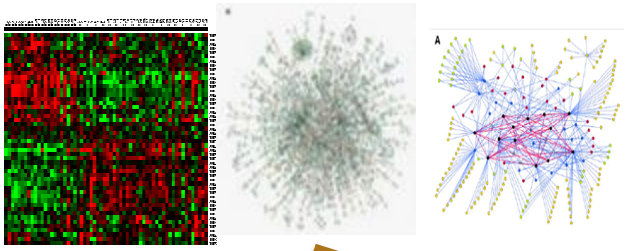
Barabasi et al., Network medicine: a network-based approach to human disease, Nature Reviews Genetics 2011

Recommender Systems in Biology

- **"What does my gene do?"**
  - **Goals:** Determine a gene's function based on who it interacts with – "**guilty-by-association**" **principle**
- **"Give me more genes like these"**
  - **Goals:**
    - Find more multiple sclerosis genes
    - Find new ciliary genes
    - Find members of a proteasome complex, etc.

"What Does My Gene Do?"

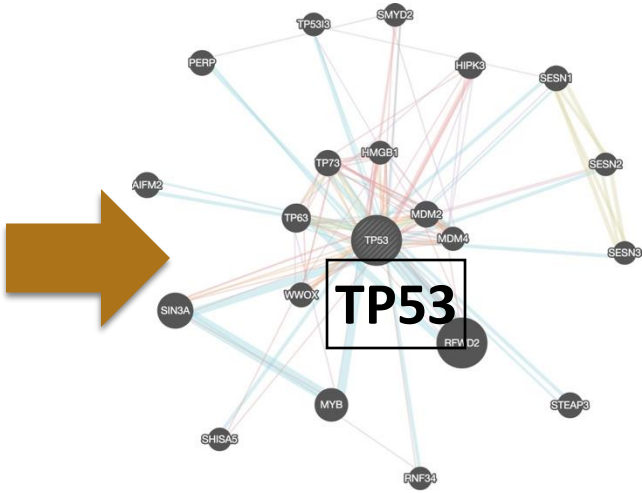
Networks



Query gene

TP53

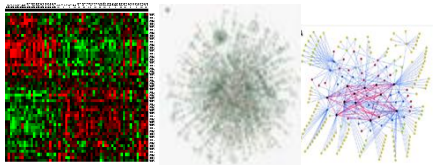
Find set ***N*** of other genes whose interactions are "**similar**" to ***TP53***'s interactions



Prediction using guilty-by-association principle: Estimate ***TP53***'s function in the cell based on functions of genes in ***N***

"Give Me More Genes Like These"

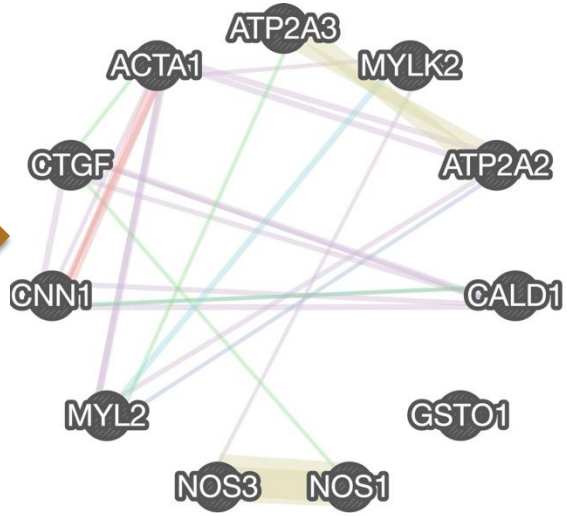
Networks



Query genes

ATP2A2
ATP2A3
NOS1
CNN1
GSTO1
NOS3
CNN2
MYLK2
CALD1
ACTA1
MYL2

Gene recommender system



Networks

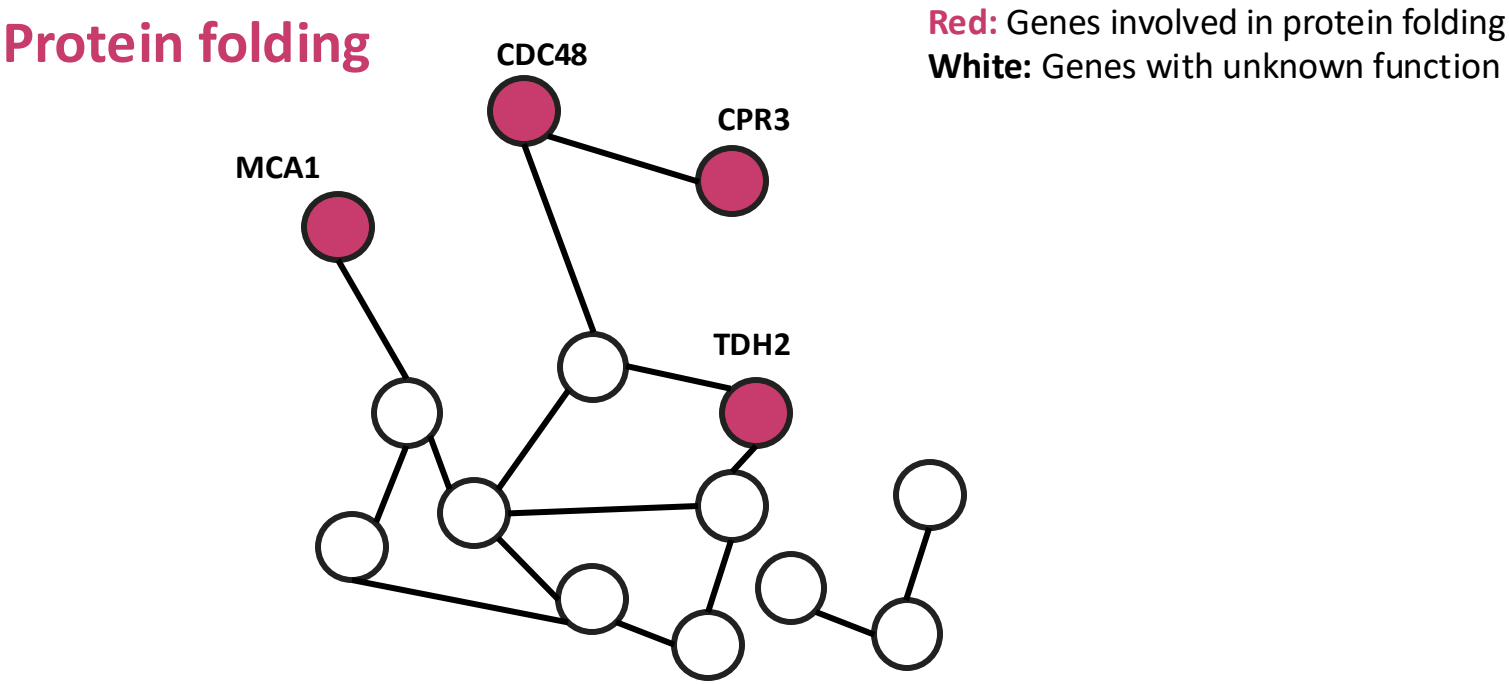
- Co-expression
- Shared protein domains
- Physical interactions
- Pathway
- Co-localization
- Genetic interactions

Functions

- muscle system process
- muscle contraction
- regulation of system process
- regulation of muscle system process
- heart contraction

Finding "Guilty Associates"

- Predict gene functions using **guilty-by-association:**



- **What other genes participate in "protein folding"?**

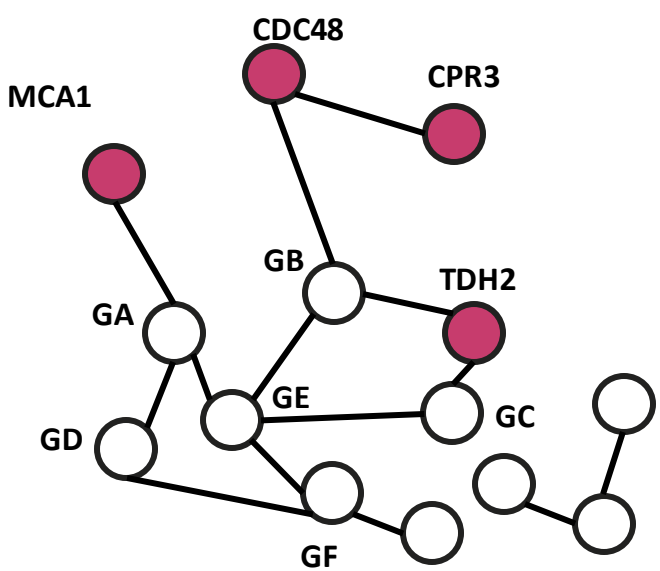
"Guilty Associates" Problem

- Let $\boldsymbol{W}$ be a $n \times n$ (weighted) adjacency matrix over n genes
- Let $\boldsymbol{y} = \{-1, 0, 1\}^n$ be a vector of **labels**:
  - 1: **positive** gene, known to be involved in a gene function/biological process
  - -1: **negative** gene
  - 0: **unlabeled** gene
- **Goal:** Predict which **unlabeled** genes are likely **positive**

"Guilty Associates" Approach

- **Approach:** Learn a vector of discriminant scores $\boldsymbol{f}$, where $\boldsymbol{f}_i$ is **likelihood** that node i is positive

- **Example:**



$$\boldsymbol{y} = [1, 1, 1, 1, 0,0,0,0,0,0,0,0,0,0]$$

$\boldsymbol{W}$ = (weighted) adjacency matrix

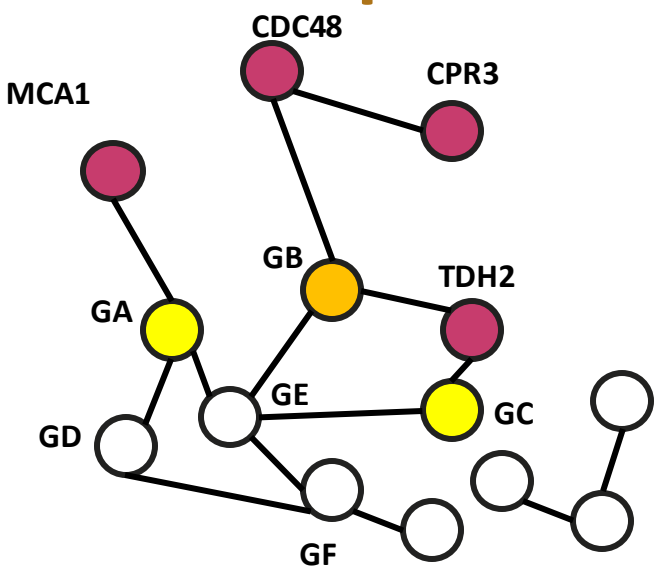
$$\boldsymbol{f} = ?$$

Approach 1: Neighbor Scoring

- Node score $\boldsymbol{f}_i$ is weighted sum of the labels of i's **direct neighbors**:

$$\boldsymbol{f}_i = \sum_{j=1}^{n} \boldsymbol{W}_{ij} \boldsymbol{y}_j$$

- **Example:**



$$\boldsymbol{f}_{\mathrm{GA}} = \boldsymbol{W}_{\mathrm{GA,MCA1}} \cdot \boldsymbol{y}_{\mathrm{MCA1}}$$

$$\boldsymbol{f}_{\mathrm{GB}} = \boldsymbol{W}_{\mathrm{GB,CDC48}} \cdot \boldsymbol{y}_{\mathrm{CDC48}} + \boldsymbol{W}_{\mathrm{GB,TDH2}} \cdot \boldsymbol{y}_{\mathrm{CDC48}}$$

$$\boldsymbol{f}_{\mathrm{GC}} = \boldsymbol{W}_{\mathrm{GC,TDH2}} \cdot \boldsymbol{y}_{\mathrm{TDH2}}$$

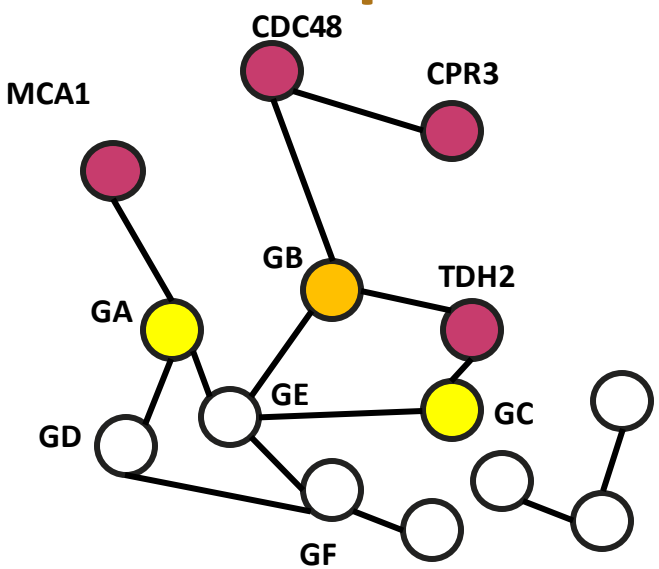
Red: Positive nodes
White: $\boldsymbol{f}_i = 0$

Approach 1: Neighbor Scoring

- Node score $\boldsymbol{f}_i$ is weighted sum of the labels of i's **direct neighbors**:

$$\boldsymbol{f}_i = \sum_{j=1}^{n} \boldsymbol{W}_{ij} \boldsymbol{y}_j$$

- **Example:**



$$\boldsymbol{f}_{\mathrm{GA}} = \boldsymbol{W}_{\mathrm{GA,MCA1}} \cdot \boldsymbol{y}_{\mathrm{MCA1}}$$

$$\boldsymbol{f}_{\mathrm{GB}} = \boldsymbol{W}_{\mathrm{GB,CDC48}} \cdot \boldsymbol{y}_{\mathrm{CDC48}} + \boldsymbol{W}_{\mathrm{GB,TDH2}} \cdot \boldsymbol{y}_{\mathrm{CDC48}}$$

$$\boldsymbol{f}_{\mathrm{GC}} = \boldsymbol{W}_{\mathrm{GC,TDH2}} \cdot \boldsymbol{y}_{\mathrm{TDH2}}$$

- **One half** of GC's neighbors are positives
- **One third** of GA's neighbors are positives
- **But:** $\boldsymbol{f}_{\mathrm{GC}} = \boldsymbol{f}_{\mathrm{GA}}$ (if $\boldsymbol{W}$ is binary)

Weighted Neighbors

- **Normalize matrix $\boldsymbol{W}$** by node degrees:

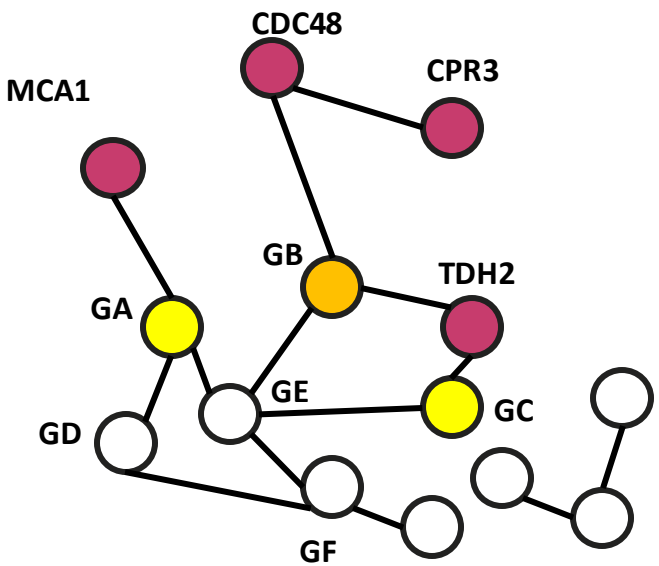
$$\boldsymbol{f}_i = \frac{1}{\boldsymbol{d}_i} \sum_{j=1}^{n} \boldsymbol{W}_{ij} \boldsymbol{y}_j \,, \qquad \boldsymbol{d}_i = \sum_j \boldsymbol{W}_{ij}$$

Matrix notation:

$$\boldsymbol{f}_i = \boldsymbol{D}^{-1} \boldsymbol{W} \boldsymbol{y}$$

$$\boldsymbol{D} = diag(\boldsymbol{d})$$

- **Example:**



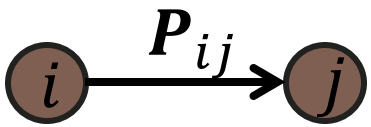
$$\boldsymbol{f}_{\mathrm{GA}} = \frac{1}{3} \boldsymbol{W}_{\mathrm{GA,MCA1}} \cdot \boldsymbol{y}_{\mathrm{MCA1}}$$

$$\boldsymbol{f}_{\mathrm{GB}} = \frac{1}{3} (\boldsymbol{W}_{\mathrm{GB,CDC48}} \cdot \boldsymbol{y}_{\mathrm{CDC48}} + \boldsymbol{W}_{\mathrm{GB,TDH2}} \cdot \boldsymbol{y}_{\mathrm{TDH2}})$$

$$\boldsymbol{f}_{\mathrm{GC}} = \frac{1}{2} \boldsymbol{W}_{\mathrm{GC,TDH2}} \cdot \boldsymbol{y}_{\mathrm{TDH2}}$$

Random Walks

- Matrix $\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{W}$ is known as **Markov transition matrix**
  - $\boldsymbol{D}$ is a diagonal matrix with diagonal elements $\boldsymbol{d}_i$
  - $\boldsymbol{P}$ is **a row stochastic matrix**, $\sum_j \boldsymbol{P}_{ij} = 1$
- Row i is a probability distribution over **random walks** starting at node i

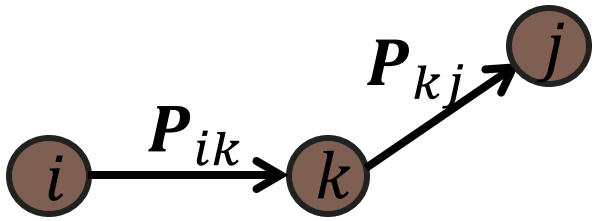


- $\boldsymbol{P}_{ij}$ is probability of a **random walker following a link from node i to node j**

Indirect Neighbor Scoring

- Use **random walks** to include **indirect neighbors** in computations
- **Idea:** Extend **direct neighbor scoring** formula $\boldsymbol{f} = \boldsymbol{D}^{-1}\boldsymbol{W}\boldsymbol{y} = \boldsymbol{P}\boldsymbol{y}$ to include **2-hop neighbors**
- Probability of a random walk of length **two** between node i and node j is:

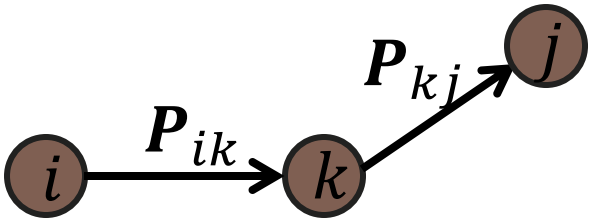
$$[\boldsymbol{P}^2]_{ij} = \sum_{k=1}^{n} \boldsymbol{P}_{ik}\boldsymbol{P}_{kj}$$



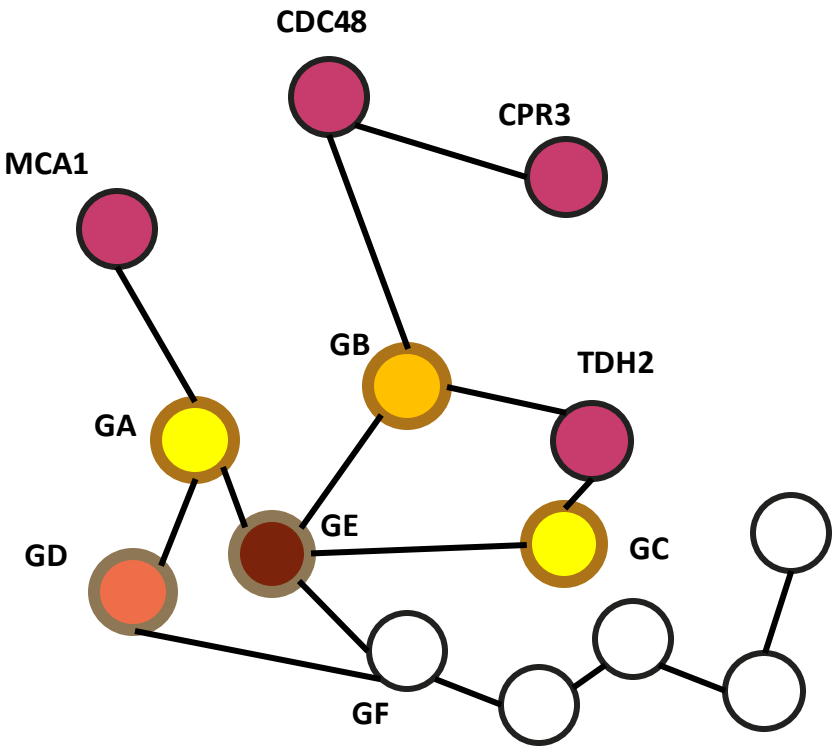
Approach 2: 2-Hop Neighbors

- Consider **2-hop neighbors** when calculating node score $\boldsymbol{f}_i$ as:

$$\boldsymbol{f}_i = \underbrace{\sum_{j=1}^{n} \boldsymbol{P}_{ij}\boldsymbol{y}_j}_{\text{Direct neighbors}} + \underbrace{\sum_{j=1}^{n} [\boldsymbol{P}^2]_{ij}\, \boldsymbol{y}_j}_{\text{2-hop neighbors}}$$



Example: 2-Hop Neighbors



$$\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{W}$$

$$\boldsymbol{f}_i = \underbrace{\sum_{j=1}^{n} \boldsymbol{P}_{ij}\boldsymbol{y}_j}_{\text{Direct neighbors}} + \underbrace{\sum_{j=1}^{n} [\boldsymbol{P}^2]_{ij}\,\boldsymbol{y}_j}_{\text{2-hop neighbors}}$$

$$\boldsymbol{f}_{\mathrm{GA}} = \boldsymbol{P}_{\mathrm{GA,MCA1}} \cdot \boldsymbol{y}_{\mathrm{MCA1}}$$

$$\boldsymbol{f}_{\mathrm{GE}} = \boldsymbol{P}^2_{\mathrm{GE,MCA1}} \cdot \boldsymbol{y}_{\mathrm{MCA1}} + \boldsymbol{P}^2_{\mathrm{GE,TDH2}} \cdot \boldsymbol{y}_{\mathrm{TDH2}} + \boldsymbol{P}^2_{\mathrm{GE,CDC48}} \cdot \boldsymbol{y}_{\mathrm{CDC48}}$$

- Direct neighbor of a positive gene
- 2-hop neighbor of a positive gene

Red: Positive genes

White: $\boldsymbol{f}_i = 0$

$[\boldsymbol{P}^2]_{ij} > 0$ if there is a walk of length 2 between i and j

Beyond 2-Hop Neighbors

- **This approach** can be **extended** to include nodes at **distance r** (usually $r < 4$):
  - $[\boldsymbol{P}^{\boldsymbol{r}}]_{ij}$ = Probability of a walk from i to j in **r steps**
- Increasing r beyond 2 sometimes results in degradation of prediction performance
  - [Chua et al., Bioinformatics 2006; Myers et al., Genome Biology 2005, Cowen et al., Nature Reviews 2017]

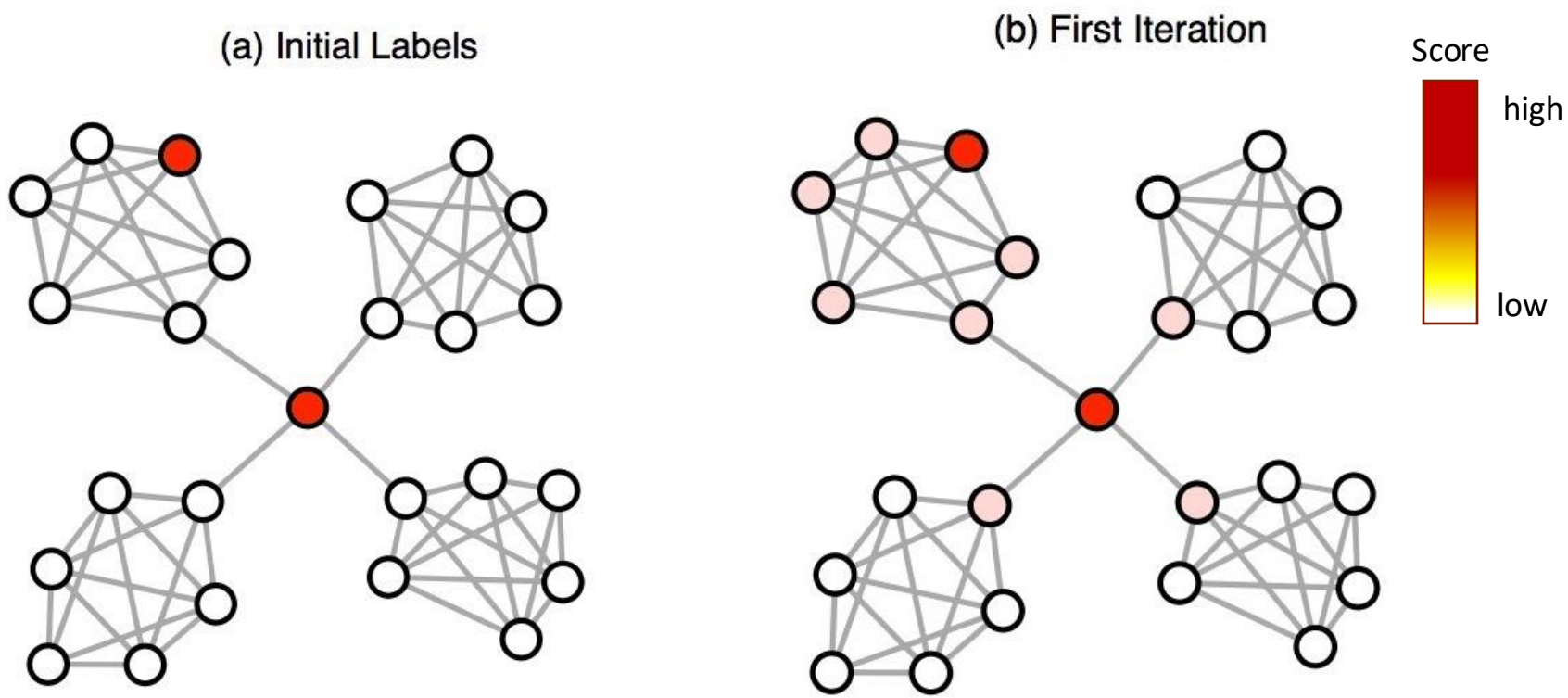
- **Next:** Use **random walks** propagate **labels** throughput the network

Beyond 2-Hops: Label Propagation

- Label propagation **generalizes** neighborhood-based approaches by **considering random walks of all possible lengths**
- The algorithm can be derived as:
  1. Iterative diffusion process [Zhou et al., NIPS 2004]
  2. Solution to a specific convex optimization task [Zhou et al., NIPS 2004, Zhu et al., ICML 2003]
  3. Maximum a posteriori (MAP) estimation in Gaussian Markov Random Fields [Rue and Held, Chapman & Hall, 2005]
- **Next:** Derivation based on **diffusion**

Label Propagation: Intuition

Intuition: Diffuse labels through edges of the network



Red: positive nodes
White: unlabeled nodes

Red: positive nodes
Pink: $f_i > 0$
White: $f_i = 0$

Diffusion Process: Idea

- **Diffusion** is defined as an **iterative process** [Zhou et al., NIPS 2004]
- **Diffuse labels through network edges:**
  - Start with initial label information, $\boldsymbol{f}_i^{(0)} = \boldsymbol{y}_i$
  - In each iteration, node i receives **label information from its neighbors** and also **retains some of its initial label**
  - λ specifies **relative amount** of label information from i's neighbors and its initial label
- Finally: Label for each unlabeled node is set to be the class (-1 or 1) of which it has **received most information**

Details

Diffusion Process: Formally

- **Diffusion process** is defined as **iteration:**
  - At iteration $r = 0$, define $\boldsymbol{f}_i^{(0)} \leftarrow \boldsymbol{y}_i$
  - At iteration $r + 1$, the score for node i is **weighted average** of the scores for i's neighbors in iteration r, and i's initial label:

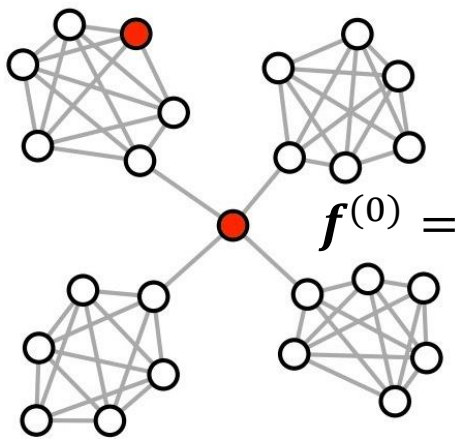
$$\boldsymbol{f}_i^{(r+1)} \leftarrow (1-\lambda)\boldsymbol{y}_i + \lambda \sum_{j=1}^{n} \boldsymbol{W}_{ij} \boldsymbol{f}_j^{(r)}$$

$0 < \lambda < 1$ is model parameter

Details

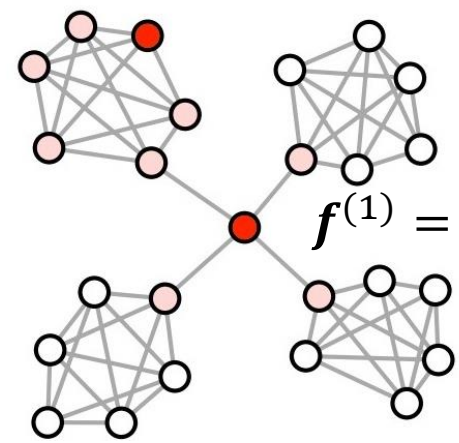
Diffusion Process: Example

(a) Initial Labels



$$\boldsymbol{f}^{(0)} = \boldsymbol{y}$$

(b) First Iteration



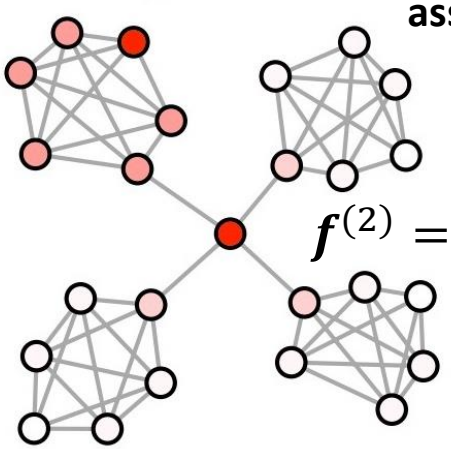
$$\boldsymbol{f}^{(1)} = \lambda \boldsymbol{W}\boldsymbol{y} + (1-\lambda)\boldsymbol{y}$$

Score

Red: positive nodes
Pink: $\boldsymbol{f}_i > 0$
White: $\boldsymbol{f}_i = 0$

(c) Second Iteration

All nodes within 2 hops are assigned a non-zero value



$$\boldsymbol{f}^{(2)} = \lambda \boldsymbol{W}\boldsymbol{f}^{(1)} + (1-\lambda)\boldsymbol{y}$$

Question: How many iterations?

Convergence Condition

- If all **eigenvalues** of $\boldsymbol{W}$ are in range $[-1, 1]$, then the sequence $\boldsymbol{f}^{(r)}$ converges to:

$$\boldsymbol{f} = (1-\lambda)\sum_{r=0}^{\infty}(\lambda \boldsymbol{W})^r \boldsymbol{y}$$

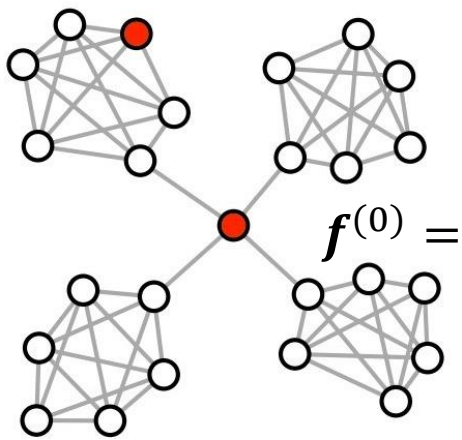
  - $[\boldsymbol{W}^r]_{ij} > 0$ if a walk of length r between i and j
  - Weight λ^r decreases with increasing distance

- $\Rightarrow$ Discriminant scores $\boldsymbol{f}$ are **weighted sum of walks of all lengths between nodes**
- $\Rightarrow$ **High value** $\boldsymbol{f}_i$: i is connected to positively labeled nodes with **many short walks**

Details

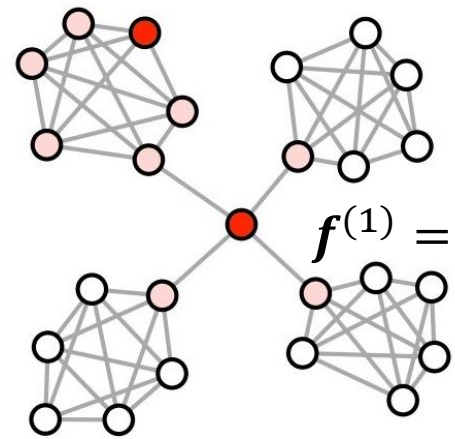
Diffusion Process: Example

(a) Initial Labels



$$\boldsymbol{f}^{(0)} = \boldsymbol{y}$$

(b) First Iteration

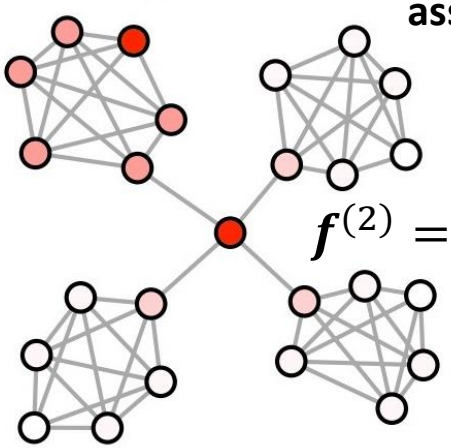


$$\boldsymbol{f}^{(1)} = \lambda \boldsymbol{W}\boldsymbol{y} + (1-\lambda)\boldsymbol{y}$$

Score

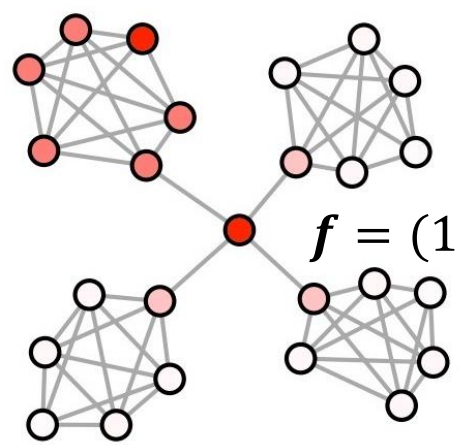
(c) Second Iteration

All nodes within 2 hops are assigned a non-zero value



$$\boldsymbol{f}^{(2)} = \lambda \boldsymbol{W}\boldsymbol{f}^{(1)} + (1-\lambda)\boldsymbol{y}$$

(d) Final Scores



$$\boldsymbol{f} = (1-\lambda)\sum_{r=0}^{\infty}(\lambda \boldsymbol{W})^r \boldsymbol{y}$$

Red: positive nodes
Pink: $\boldsymbol{f}_i > 0$
White: $\boldsymbol{f}_i = 0$

Details

Does the Process Always Converge?

- **Problem:** The infinite sum converges only if all eigenvalues of $\boldsymbol{W}$ are in $[-1, 1]$, i.e., $\rho(\boldsymbol{W}) \leq 1$
- **Solution:** Normalize $\boldsymbol{W}$ before diffusion:
  - **Symmetric** normalization:

$$\boldsymbol{S} = \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$$

$\boldsymbol{D} = diag(\boldsymbol{d})$

  - Signal is spread in a **breadth-first search** manner
  - **Asymmetric** normalization:

$$\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{W}$$

Details

Exact Solution at Convergence

- If $\rho(\boldsymbol{W}) \leq 1$, use **Taylor expansion** to compute **exact solution for label propagation:**

$$\boldsymbol{f} = (1-\lambda)\sum_{r=0}^{\infty}(\lambda \boldsymbol{S})^r \boldsymbol{y}$$

$$\Downarrow$$

$$\boldsymbol{f} = (1-\lambda)(\boldsymbol{I}-\lambda \boldsymbol{S})^{-1}\boldsymbol{y}$$

Taylor expansion, sum of geometric series:

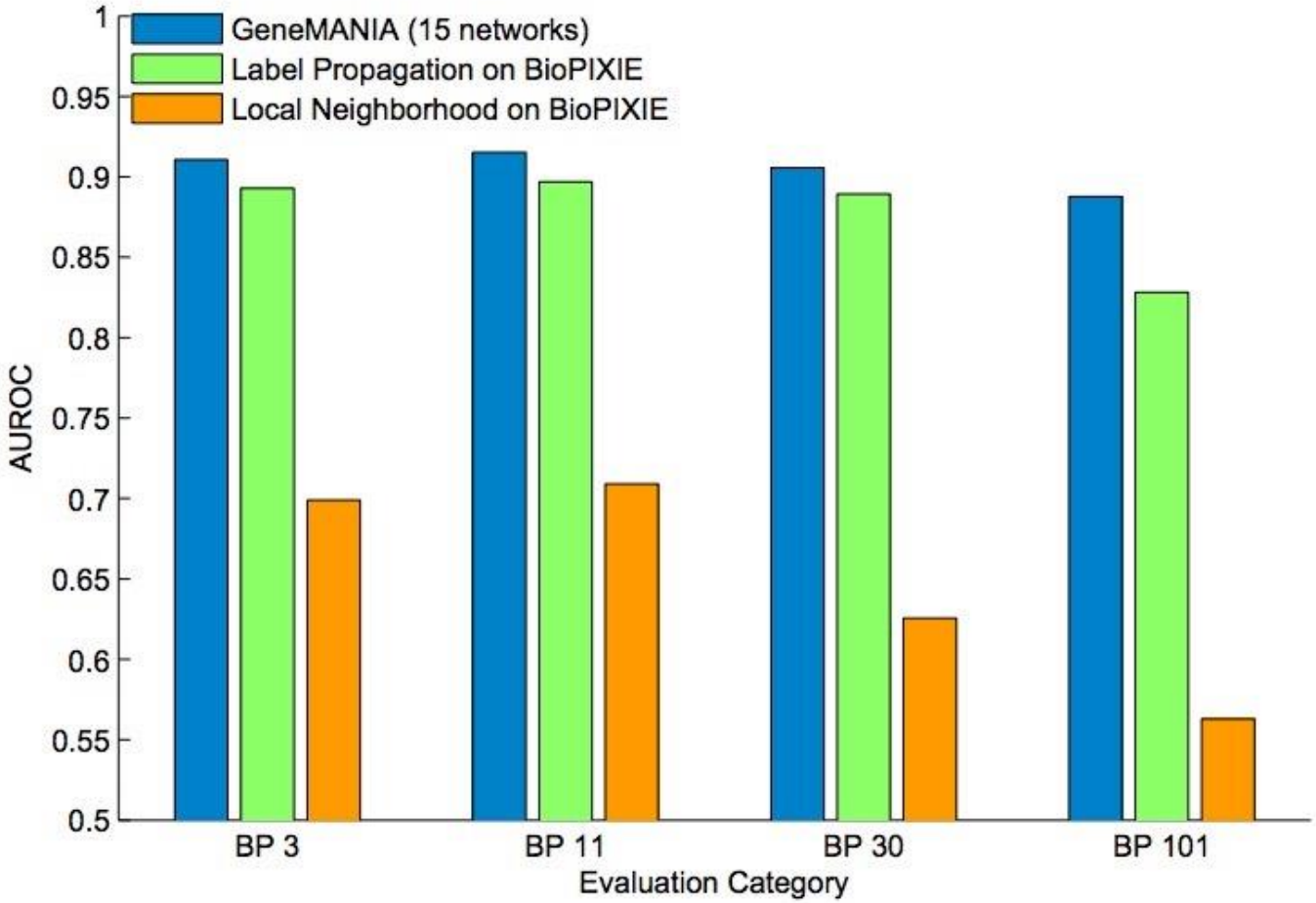
$$(I-\boldsymbol{A})^{-1} = \sum_{r=0}^{\infty} \boldsymbol{A}^r$$

Function Prediction: Setup

- **Multi-label node classification:** Node (gene) has 0+ labels (functions):
  1. For each label learn a **separate vector $\boldsymbol{f}$**:
     - **High value of $\boldsymbol{f}_i$**: i is connected to many labeled nodes through **many short walks → $\boldsymbol{i}$ likely has the label**
  2. Train: Observe a fraction of nodes and their labels
  3. Test: Predict functions for the remaining nodes
- Select optimal value for λ using **cross-validation**

[Mostafavi et al., *Genome Biology* 2008]

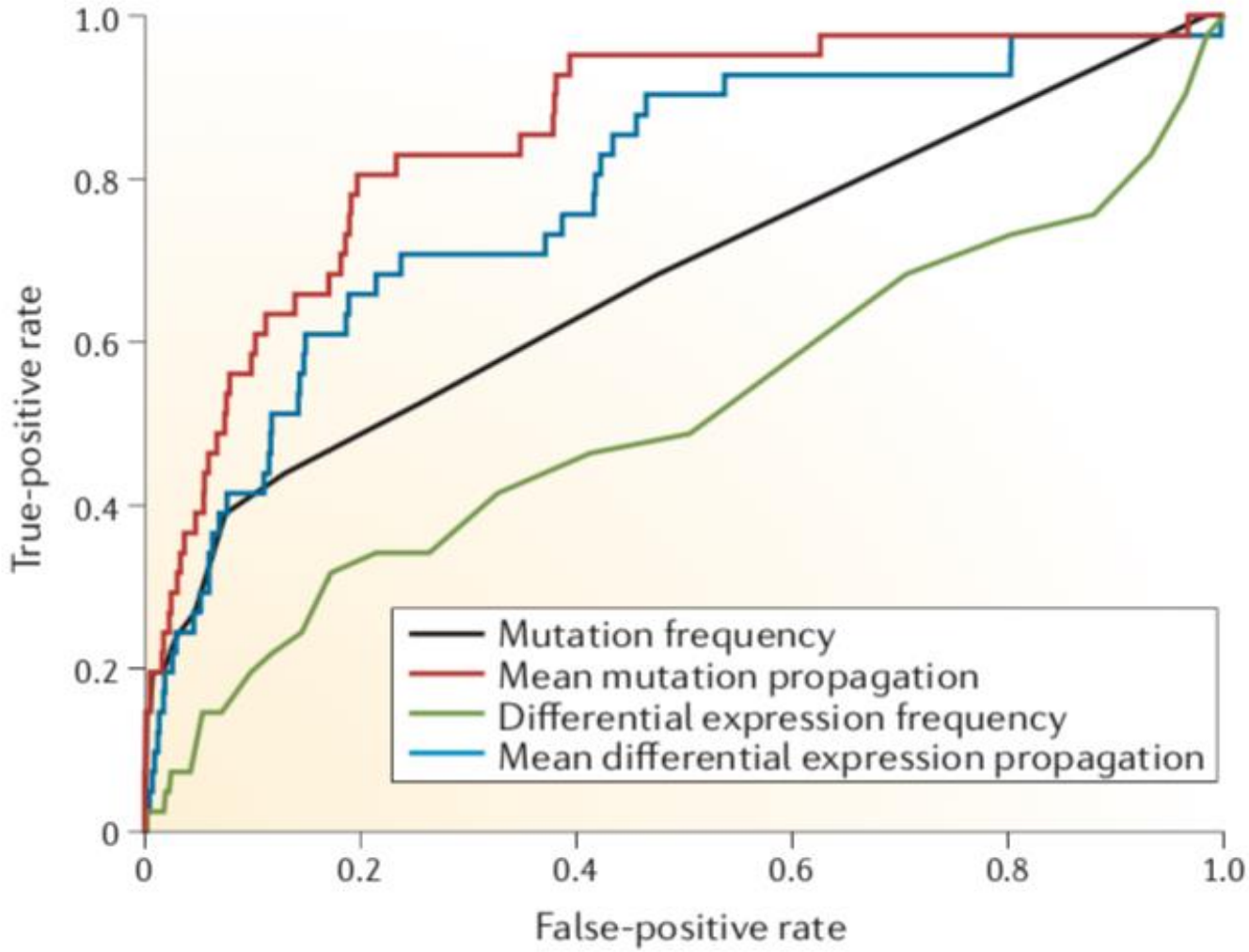
Function Prediction: Results



Label propagation outperforms neighborhood scoring methods

[Cowen et al., *Nature Reviews* 2017]

Function Prediction: Results

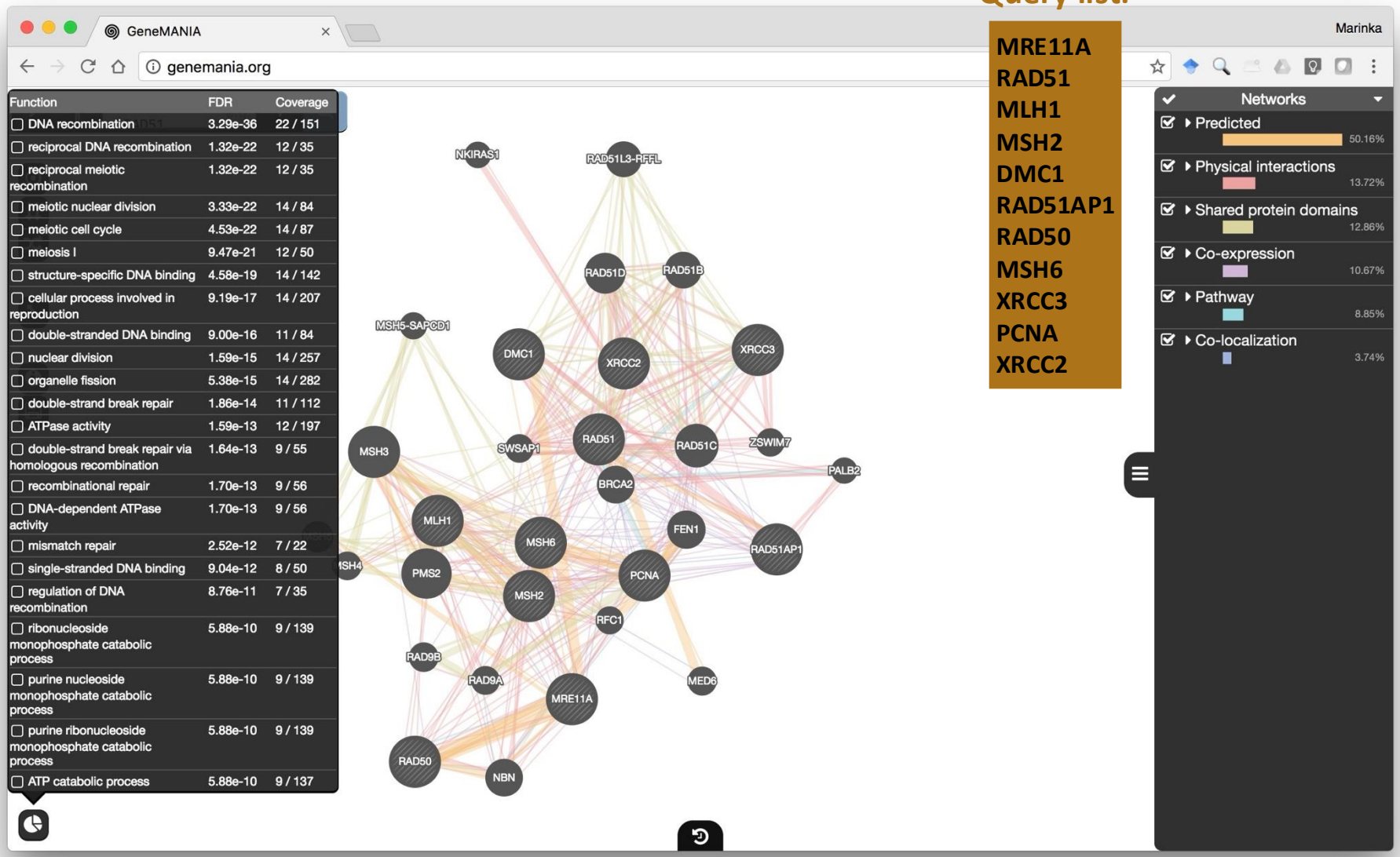


Network propagation variants outperform their frequency-based counterparts (compare the **blue** curve to the **green** curve, and the **red** curve to the **black** curve)

GeneMANIA Tool (genemania.org)

Query list:

MRE11A
RAD51
MLH1
MSH2
DMC1
RAD51AP1
RAD50
MSH6
XRCC3
PCNA
XRCC2



Quick Check

https://forms.gle/iwjeypcTrCBrkDcm7

AIM II: Artificial Intelligence in Medicine II

Artificial Intelligence in Medicine II, Spring 2025

Lecture 8: Foundations of network biology and medicine, Foundations of graph AI, Semi-supervised learning and label diffusion, Graph representation learning, Introduction to graph neural networks (GNNs), Neural message-passing models, Applications in gene function prediction, medical diagnosis, drug combination modeling, and antibiotic discovery

Course website and slides: https://zitniklab.hms.harvard.edu/AIM2

* Indicates required question

First and last name *

Your answer

Harvard email address *

Your answer

Think of another network example in biology or medicine that was not covered in today's lecture. What are nodes? How are edges defined? What predictive or generative tasks can be meaningfully defined on your network? *

Your answer

In class, we introduced the guilty-by-association approach (i.e., direct neighbor scoring, indirect neighbor scoring, label propagation) through gene function prediction. Can you think of a different biomedical problem where the same approach can be helpful? *

Your answer

Submit

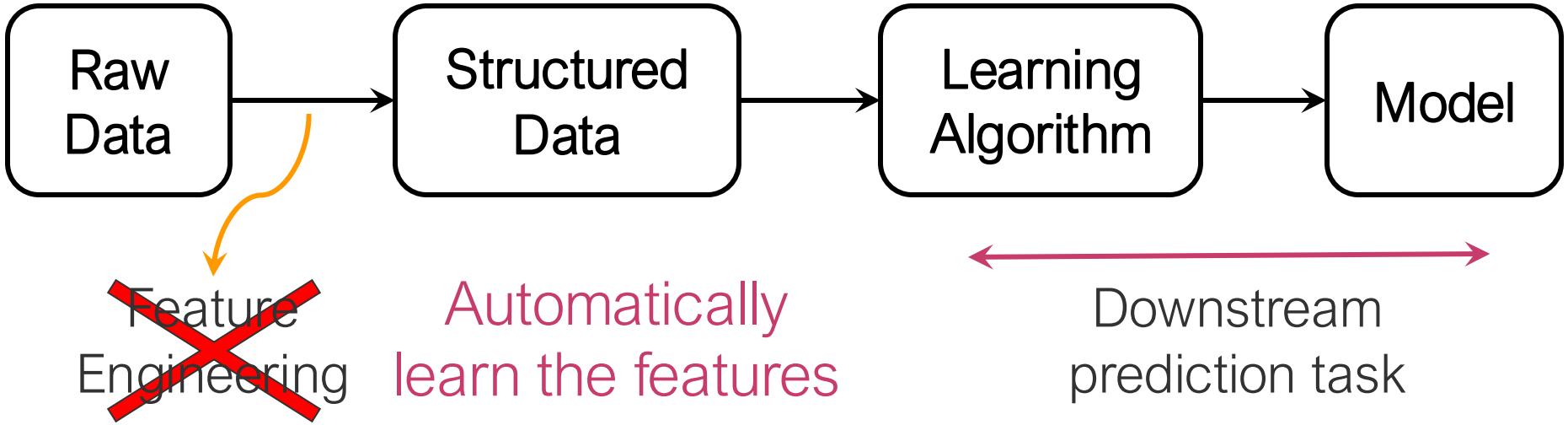
Clear form

Graph representation learning

Introduction to graph neural networks, and neural message passing models

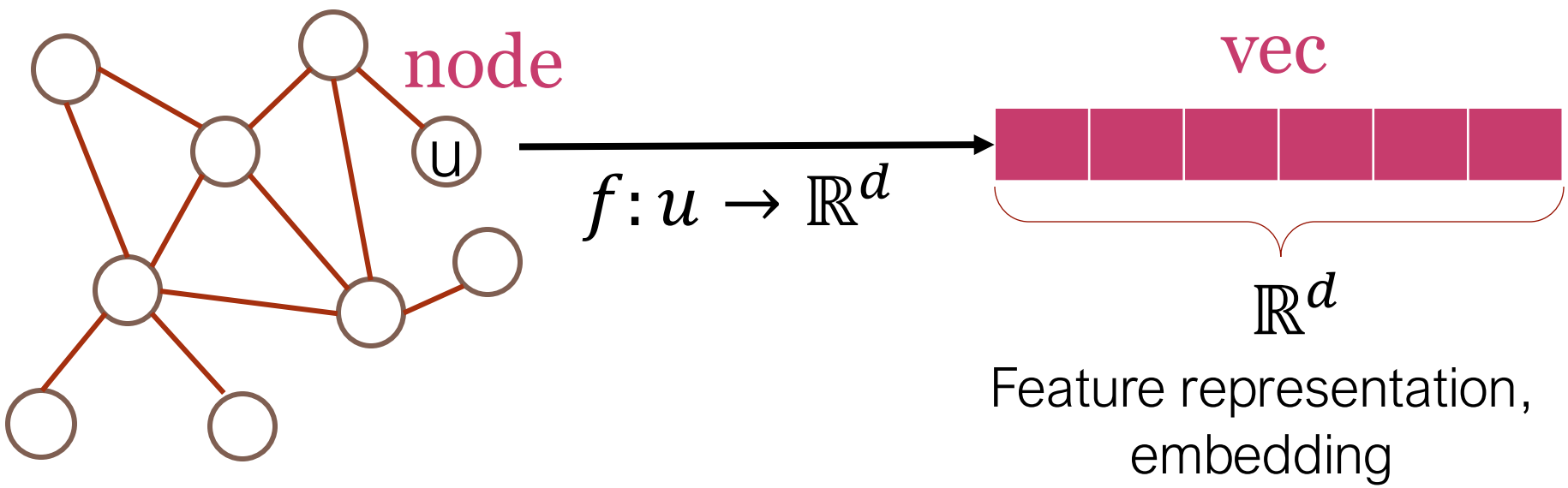
Predictive Modeling Lifecycle

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

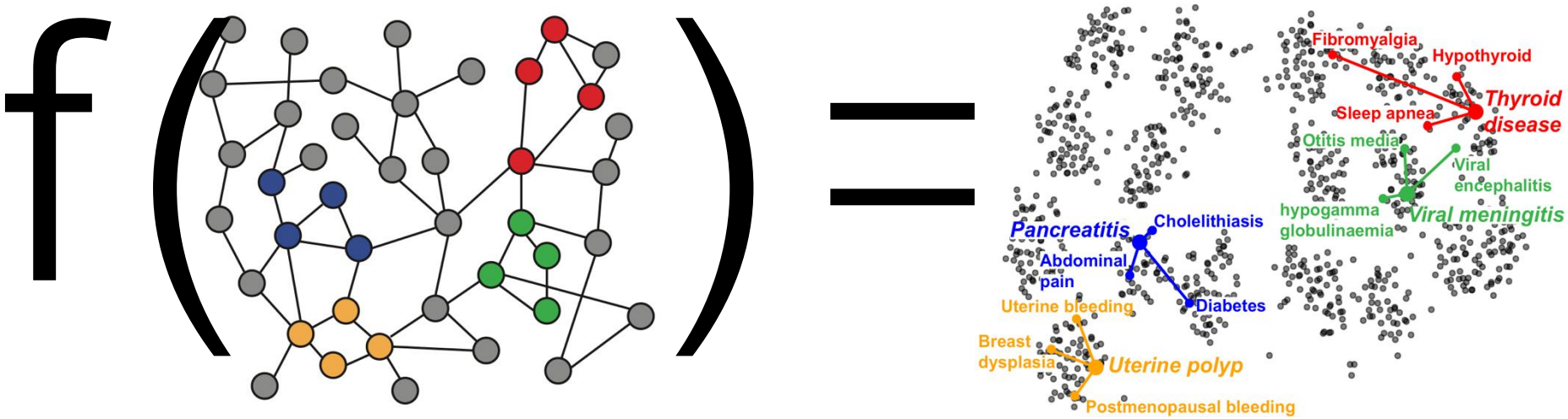


Feature Learning in Graphs

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



Embedding Nodes



Disease similarity network

Input

2-dimensional node embeddings

Output

How to learn mapping function f?

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

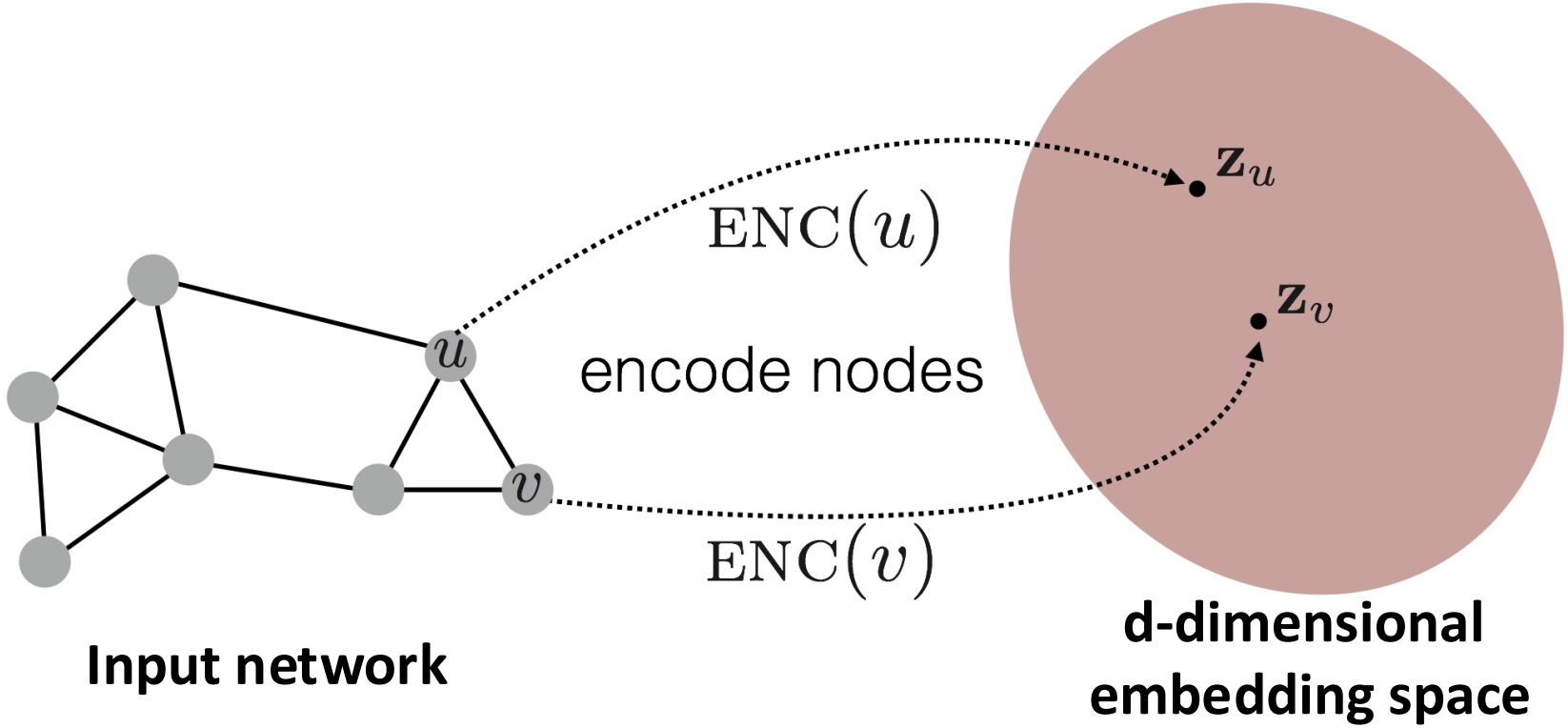
Setup

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

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

Embedding Nodes

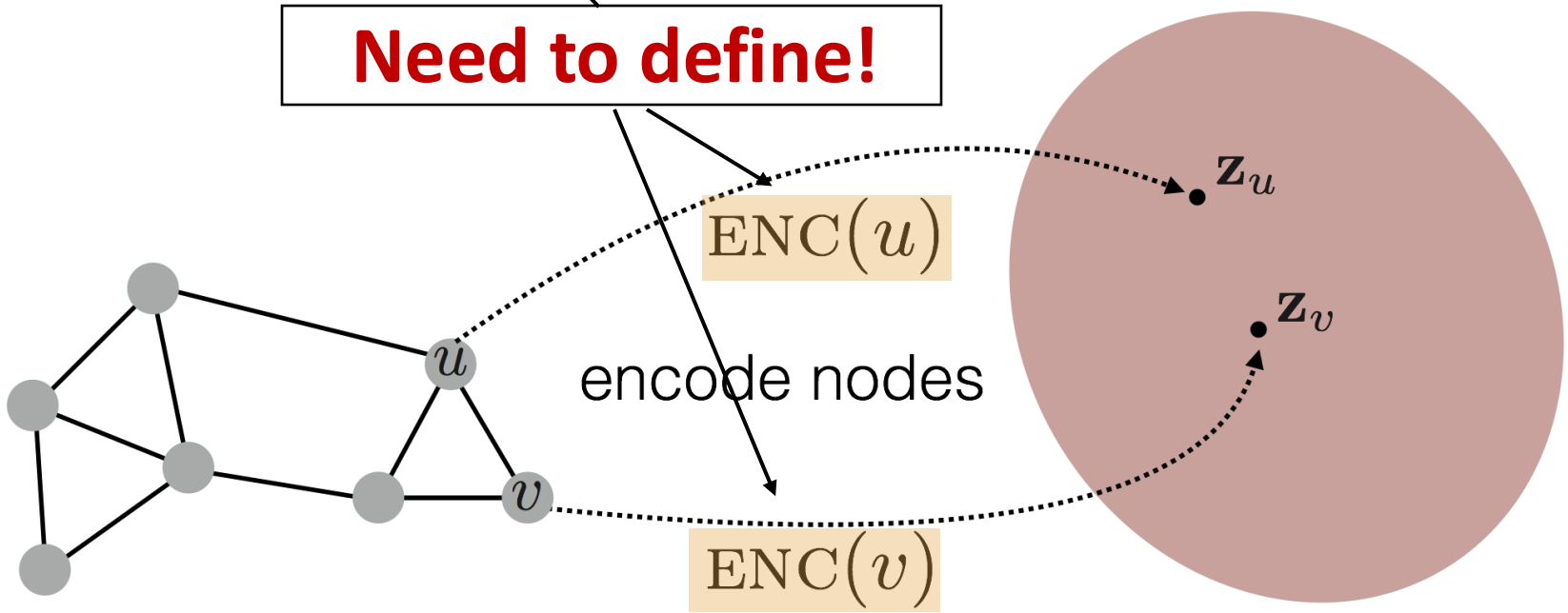
Goal: Map nodes so that similarity in the embedding space (e.g., dot product) approximates similarity in the network



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 $\boldsymbol{z}_u$)
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:

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

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

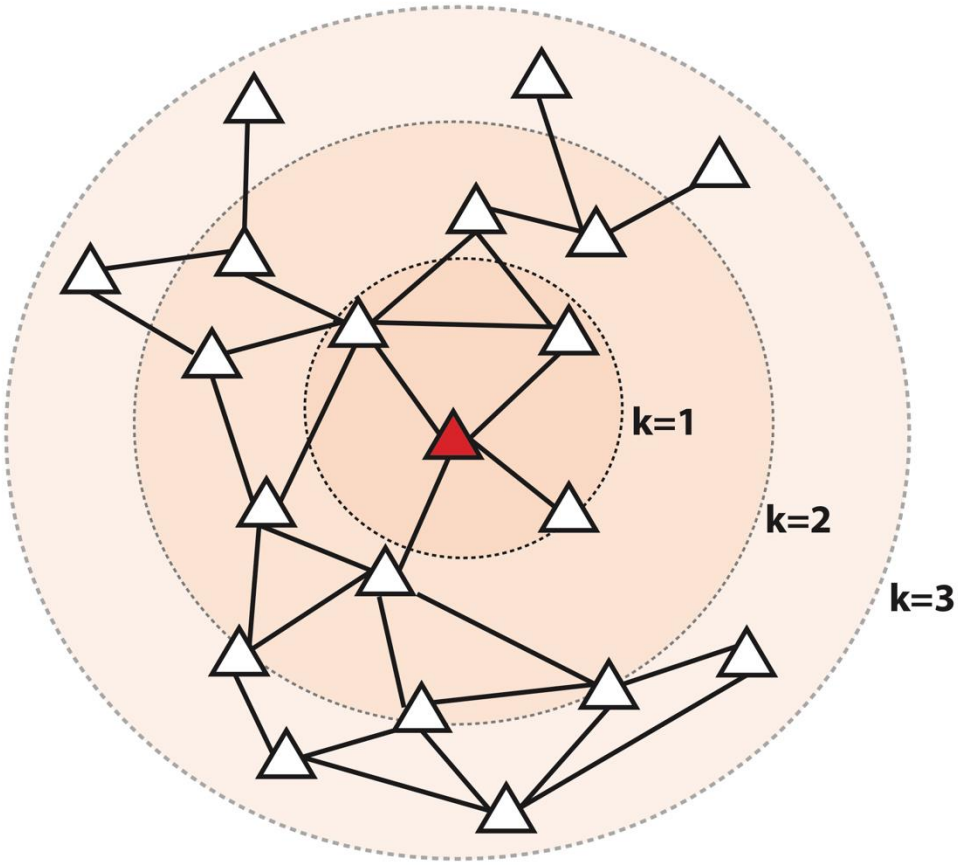
- These methods use different notions of node similarity:
  - Two nodes have similar embeddings if:
    - they are connected?
    - they share many neighbors?
    - they have similar local network structure?
    - etc.

Shallow graph representation learning

Node2vec: 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:

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

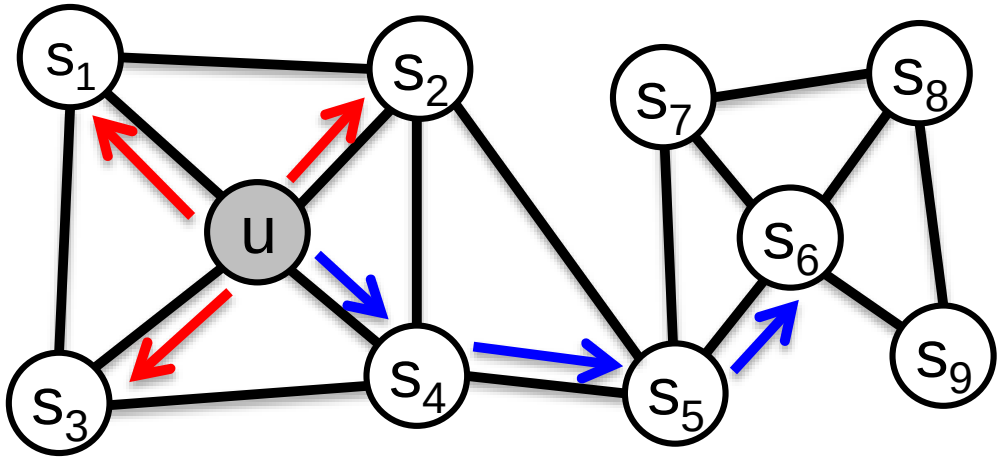
Random-Walk Embeddings

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

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 $\mathbf{L}$

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 $\mathrm{N}_R(u)$:

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

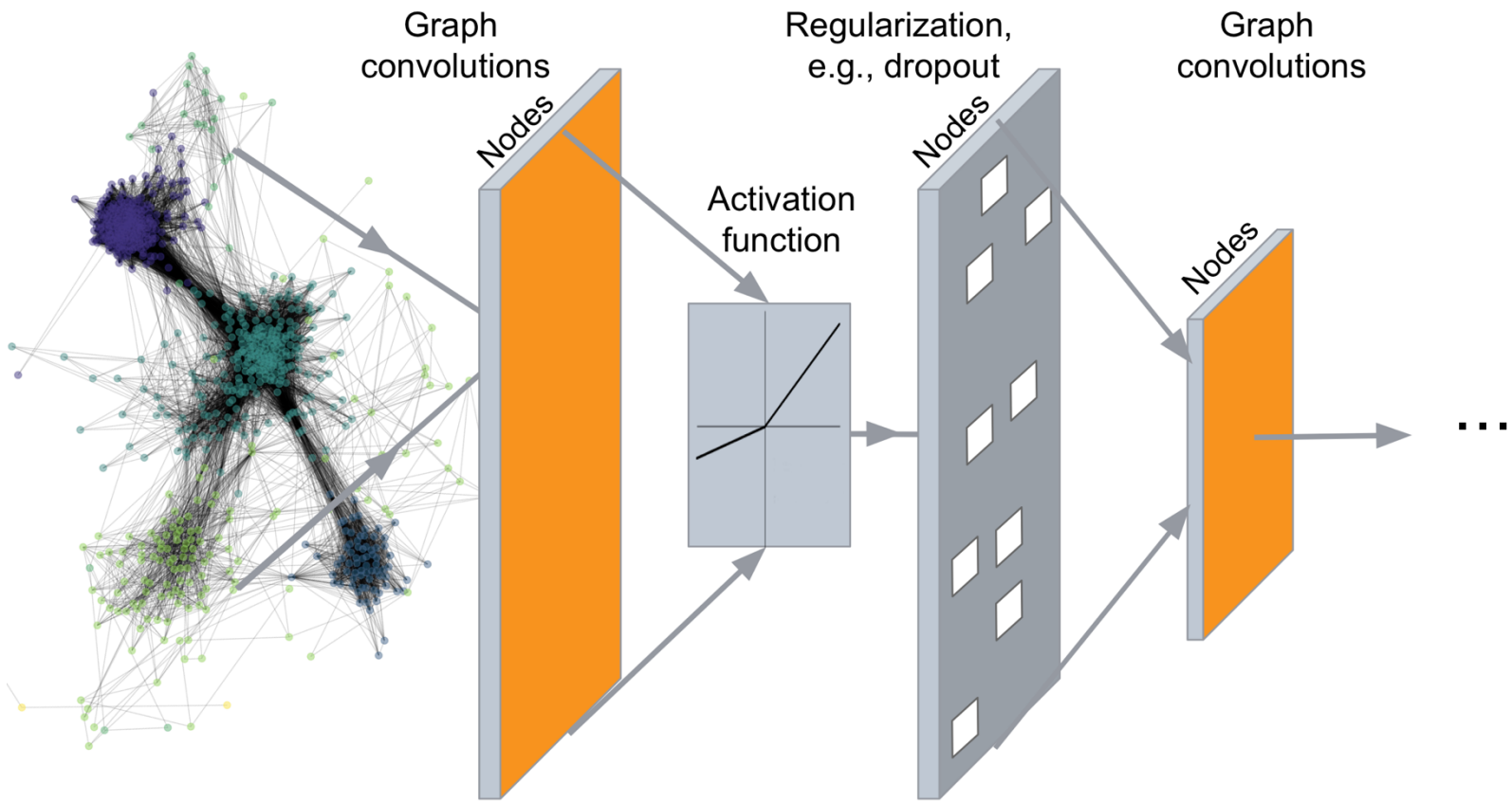
Can efficiently approximate using negative sampling

Deep graph representation learning

GNNs and neural message passing

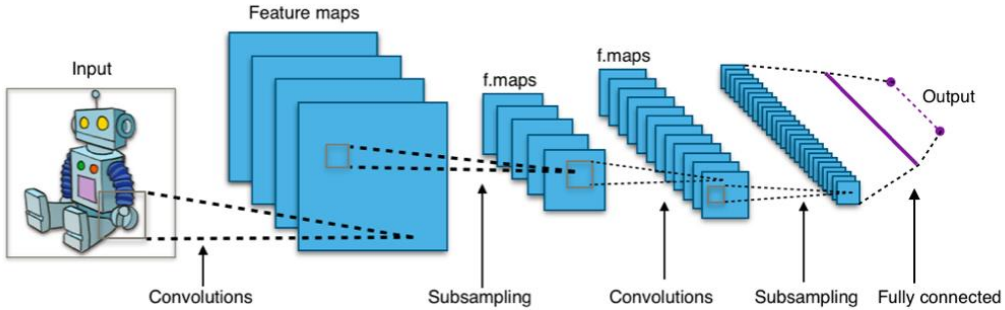
Graph neural networks

- Encoder: Multiple layers of nonlinear transformation of graph structure

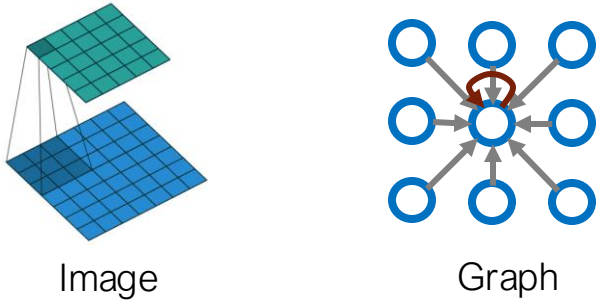


Convolutional networks

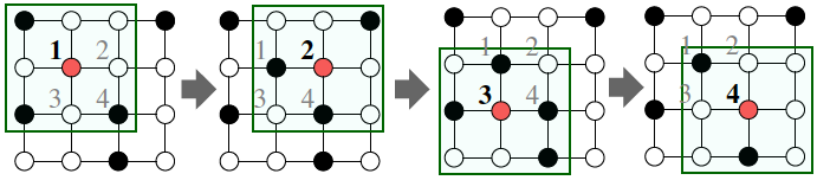
- Let's start with convolutional networks on an image:



- Single convolutional network with a 3x3 filter:

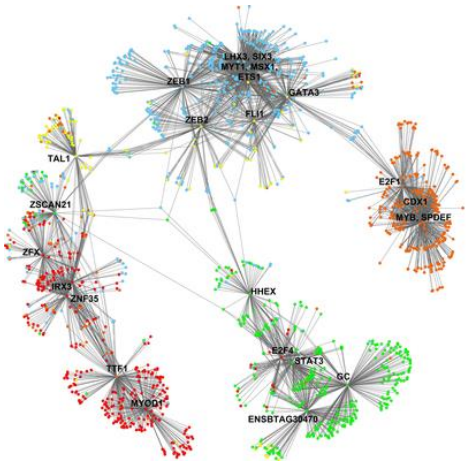


- Transform information (or messages) from the neighbors and combine them: $\sum_i W_i \, h_i$

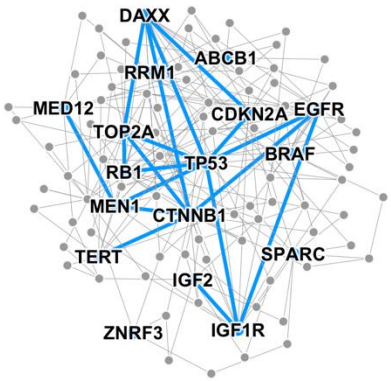


Real world graphs

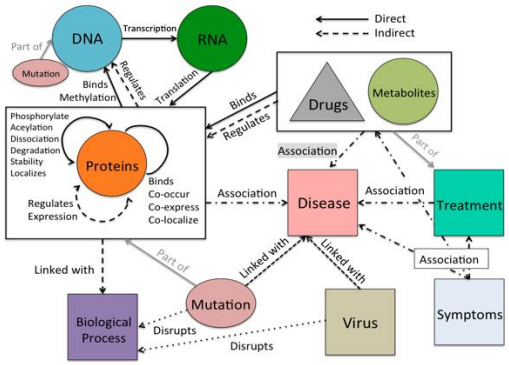
- But what if your graphs look like this?



Gene interaction network



Disease pathways

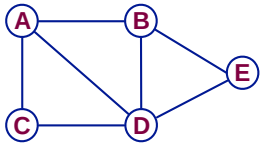


Biomedical knowledge graphs

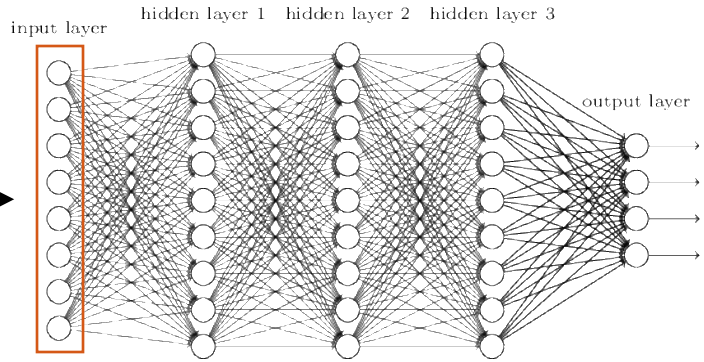
- Examples:
  - Biological or medical networks
  - Social networks
  - Information networks
  - Knowledge graphs
  - Communication networks
  - Web graphs
  - …

Naïve approach

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



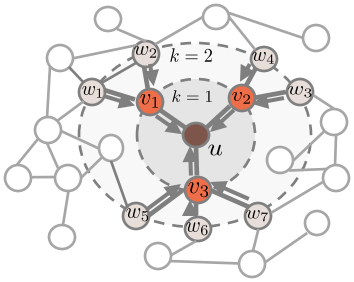
	A	B	C	D	E	Feat	
A	0	1	1	1	0	1	0
B	1	0	0	1	1	0	0
C	1	0	0	1	0	0	1
D	1	1	1	0	1	1	1
E	0	1	0	1	0	1	0



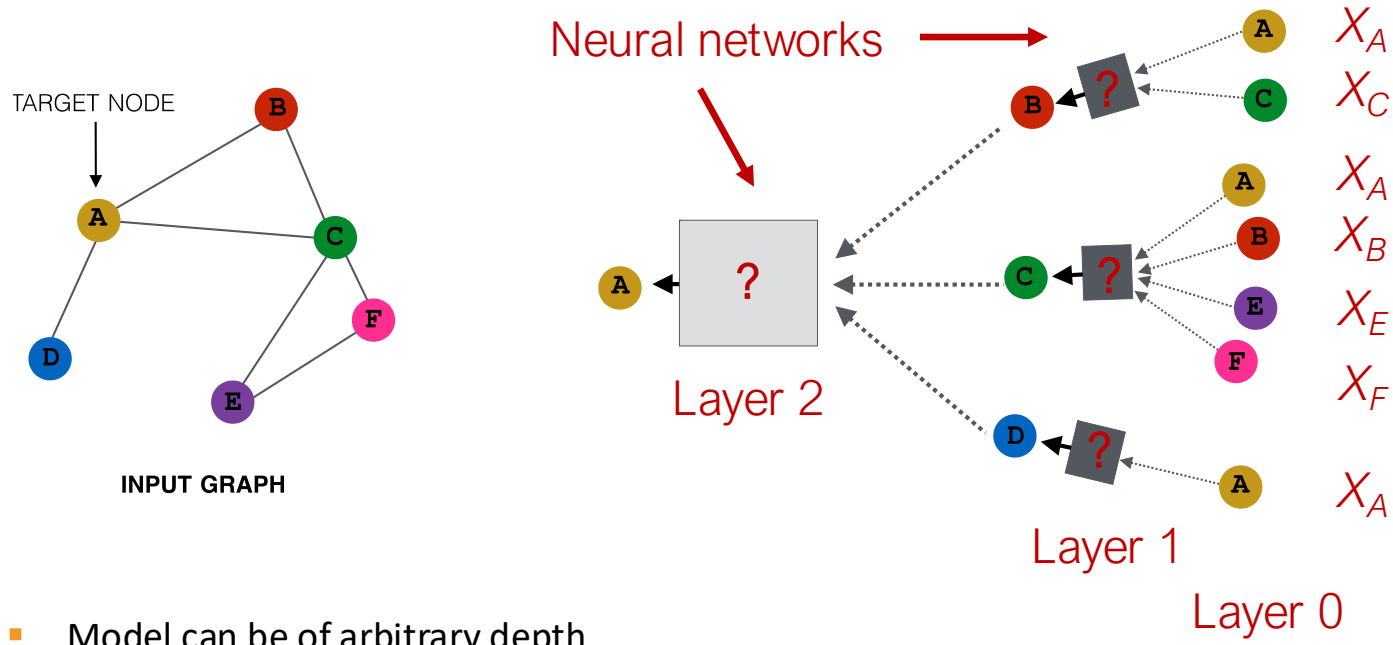
?

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

Graph neural networks

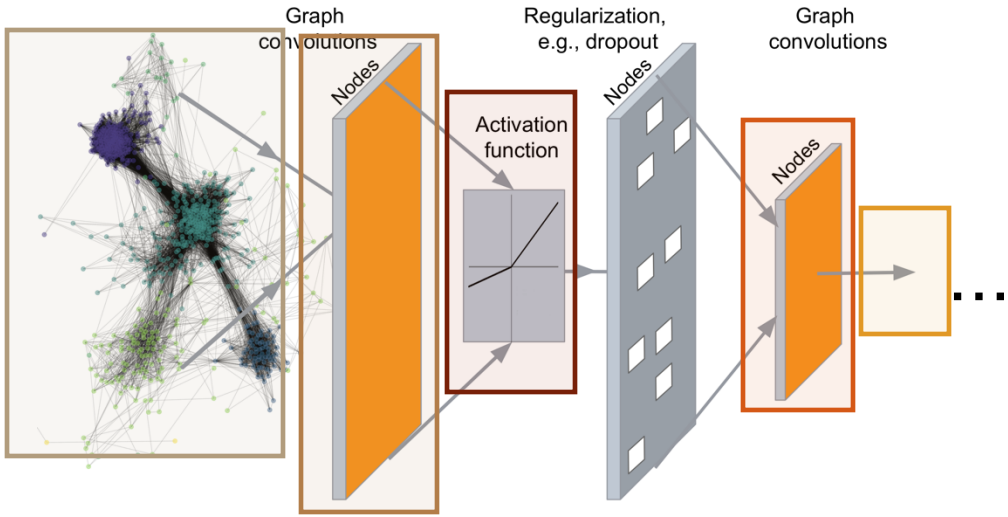


- Intuition:
  - Each node's neighborhood defines a computational graph
  - Generate node embeddings based on local network neighborhoods
- Neighborhood aggregation:



  - 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

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

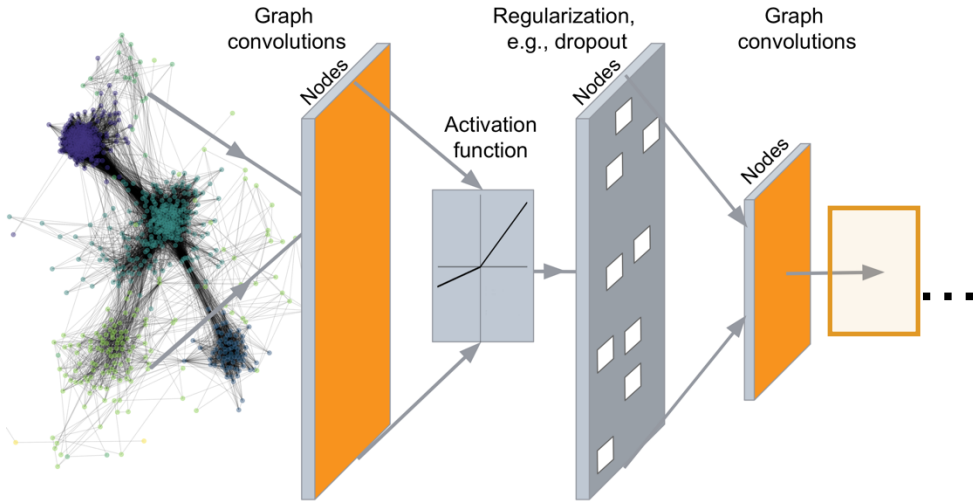
$$\mathbf{z}_v = \mathbf{h}_v^K$$

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)

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

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

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

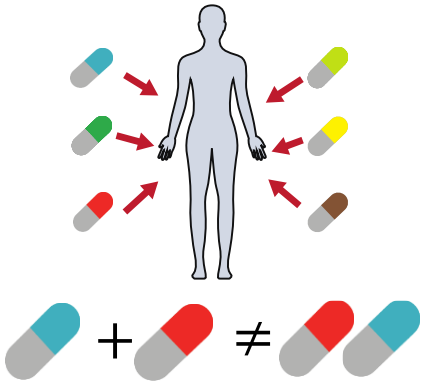
We can feed these into any loss function and run stochastic gradient descent to train the weight parameters

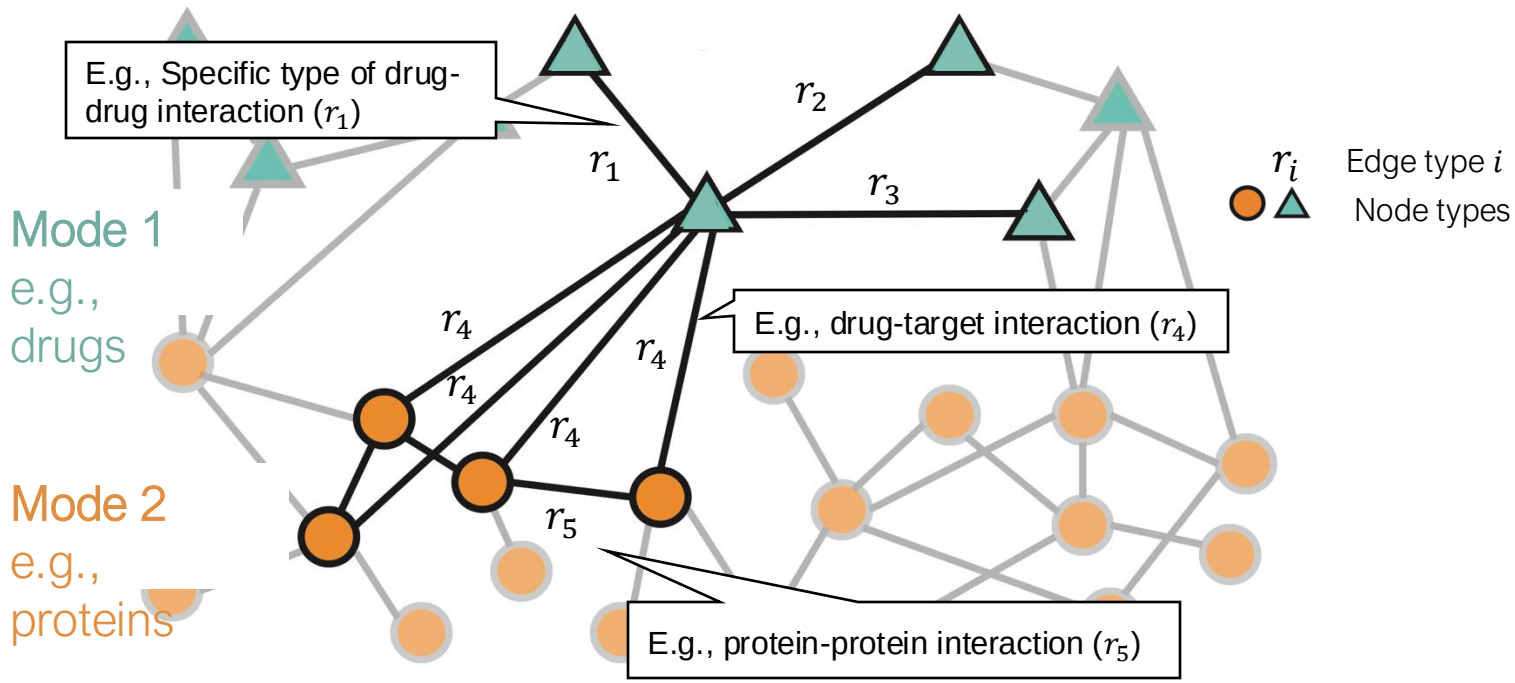
Applications in polypharmacy and drug design

Drug combination modeling, antibiotic discovery

Application: Drug combinations

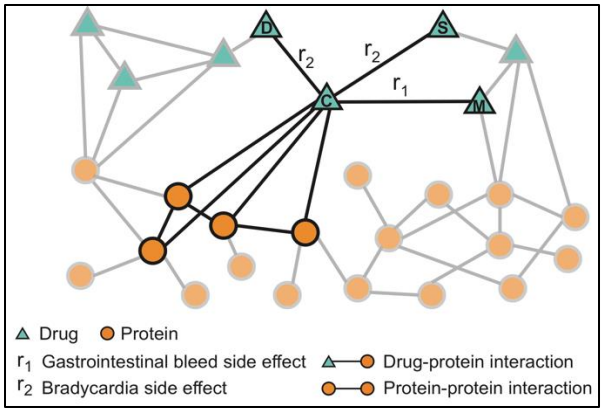
- Combinatorial explosion
  - >13 million possible combinations of 2 drugs
  - >20 billion possible combinations of 3 drugs
- Non-linear & non-additive interactions
  - Different effect than the additive effect of individual drugs
- Small subsets of patients
  - Side effects are interdependent
  - No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





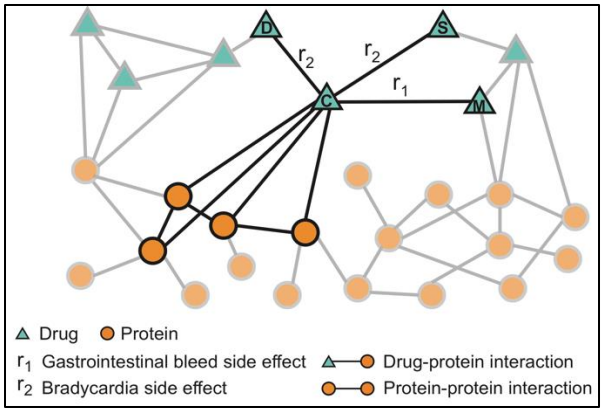
Modeling Polypharmacy Side Effects with Graph Convolutional Networks, Bioinformatics, 2018

Polypharmacy dataset

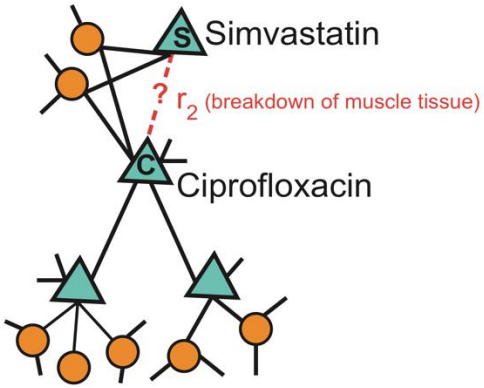


- Molecular, drug, and patient data for all US-approved drugs
  - 4,651,131 drug-drug edges: Patient data from adverse event system, tested for confounders [FDA]
  - 18,596 drug-protein edges
  - 719,402 protein-protein edges: Physical, metabolic enzyme-coupled, and signaling interactions
  - Drug and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

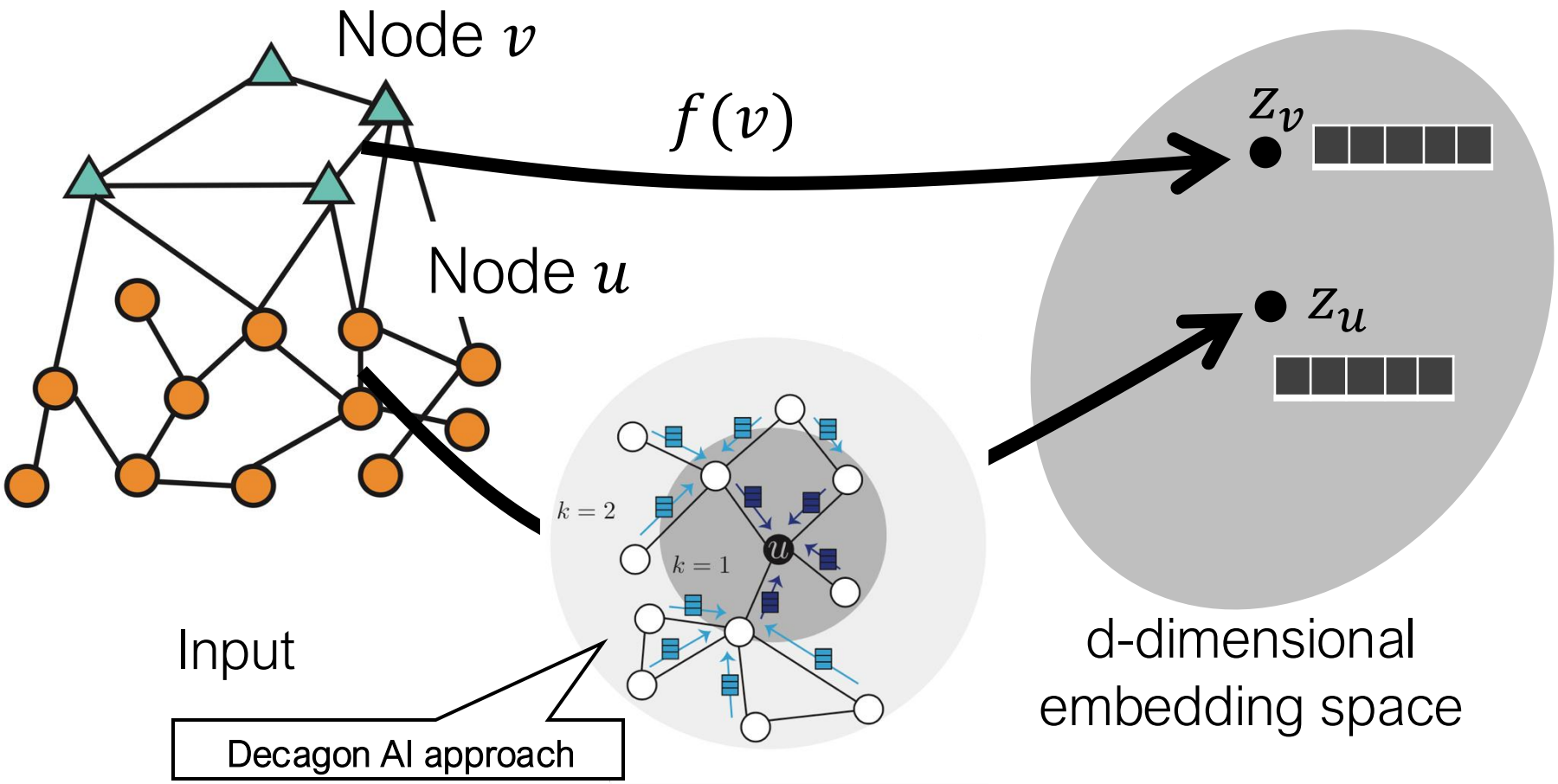


- Two main stages:
  1. Learn an embedding for every node in polypharmacy network
  2. Predict a score for every drug-drug, drug-protein, protein-protein pair in the test set based on the embeddings



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

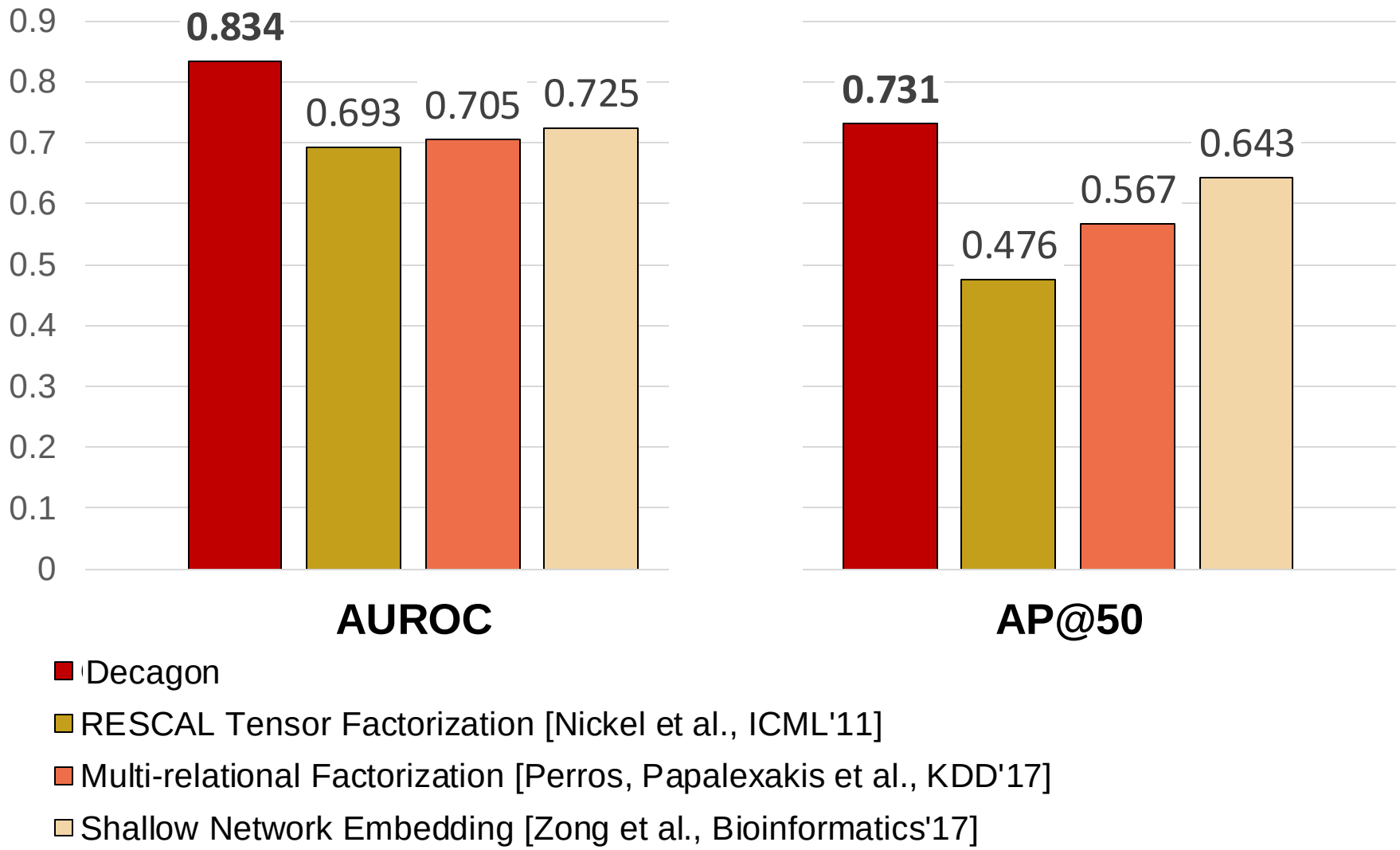
Approach: Graph Neural Network



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

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

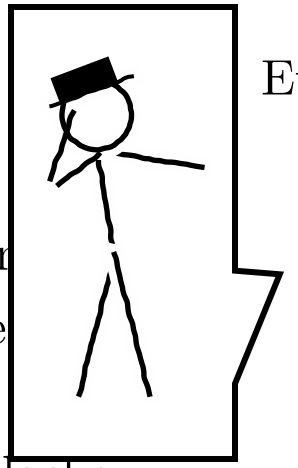
Results: Polypharmacy side effects



Results: Polypharmacy side effects

Approach:

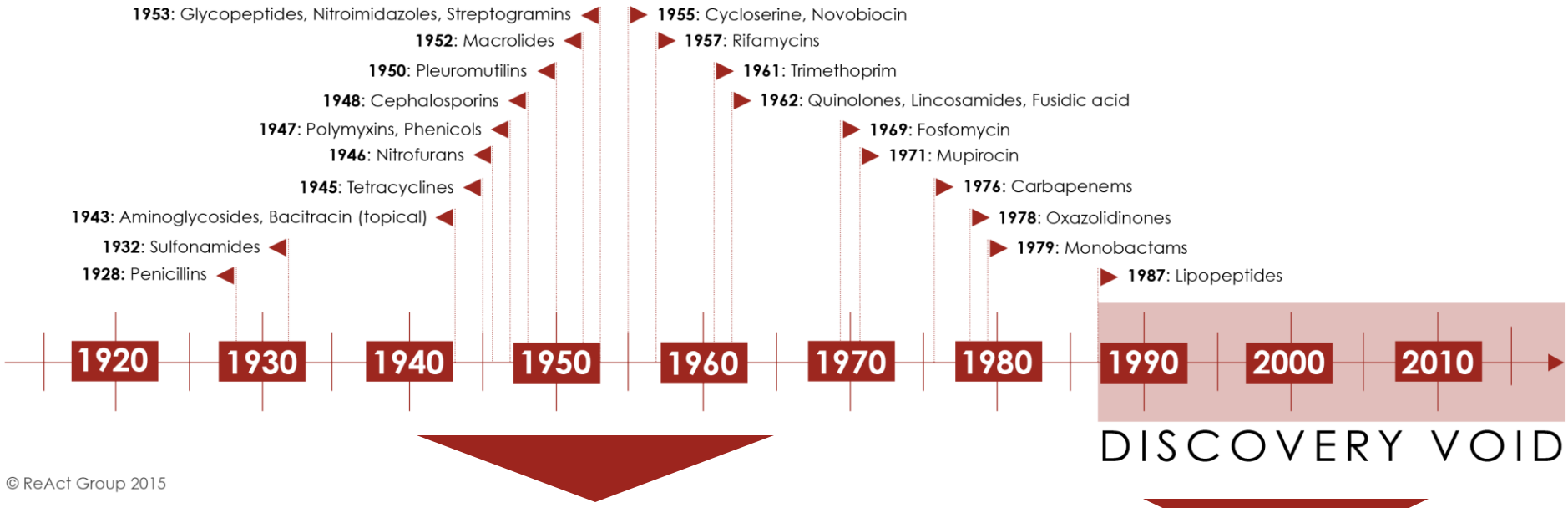
1) Train deep model on data generated **prior to 2012**
2) How many **predictions** have been **confirmed after 2012**?



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	Anagr	Azelaic acid	Cerebral thrombosis	
8	Atorvastatin	Amlodipine	Muscle inflammation	
9	Aliskiren	Tioconazole	Breast inflammation	
10	Estradiol	Nadolol	Endometriosis	

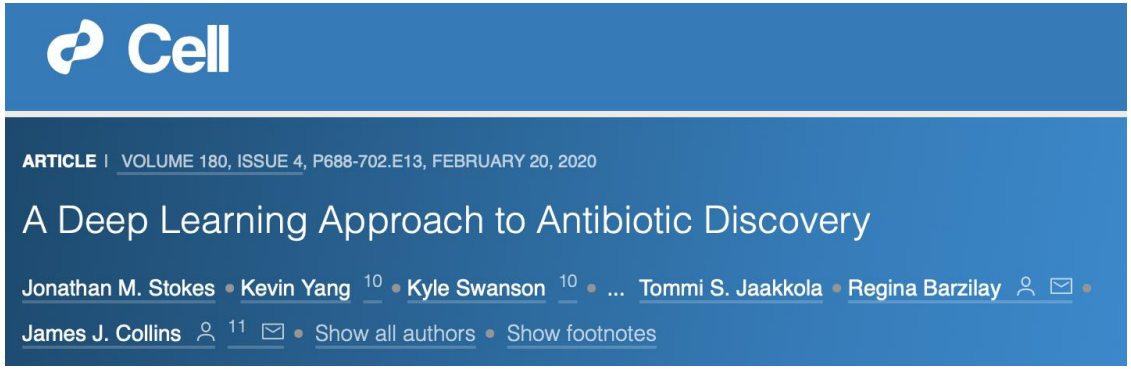
Case Report
Severe Rhabdomyolysis due to Presumed Drug Interactions between Atorvastatin with Amlodipine and Ticagrelor

Application: Antibiotic discovery

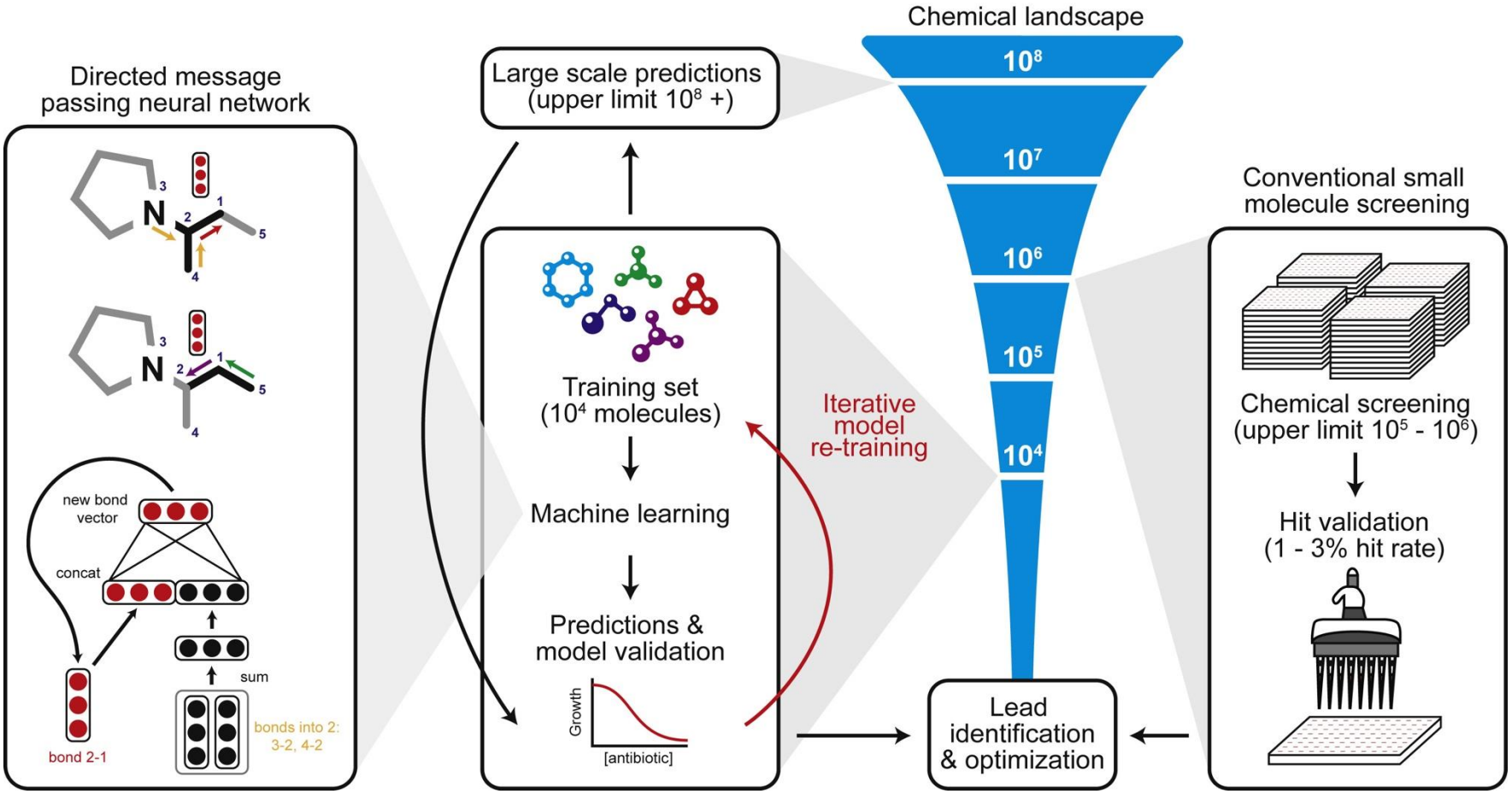


Natural product mining

Small compound screening



GNNs to learn molecular structure

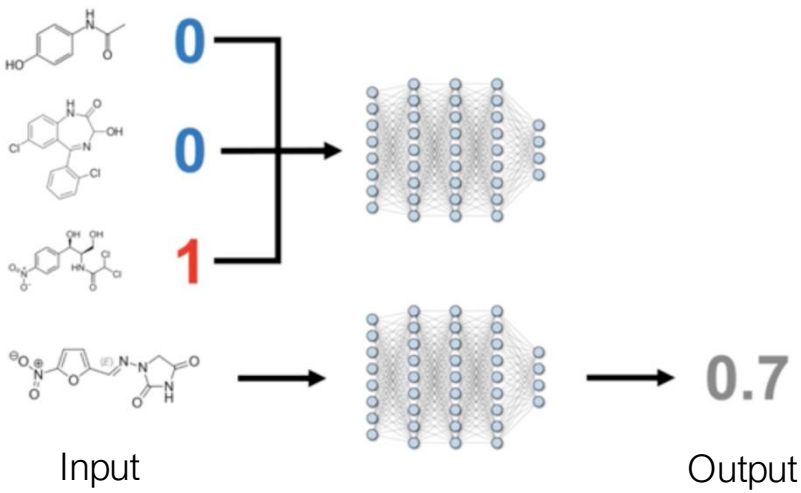


Directed message passing neural network model iteratively (1) learns representations of molecules and (2) optimizes the representations for predicting growth inhibition

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

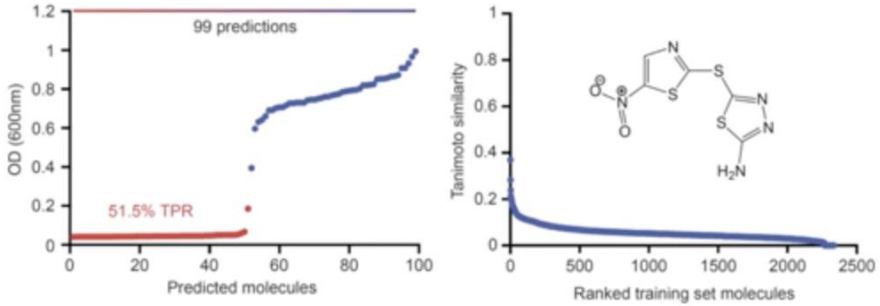
Experimental setup

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

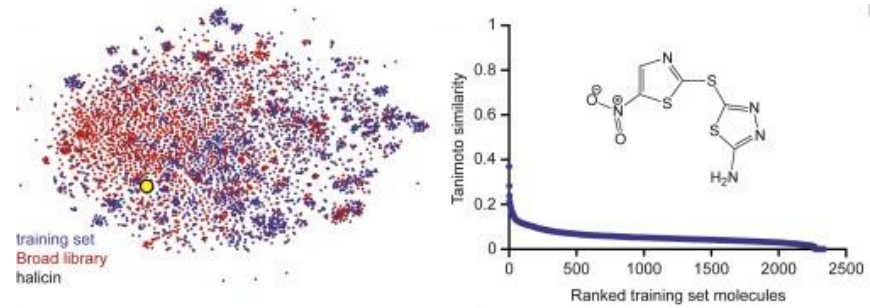
Task: Test top 99 predictions & prioritize based on similarity to known antibiotics or predicted toxicity

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

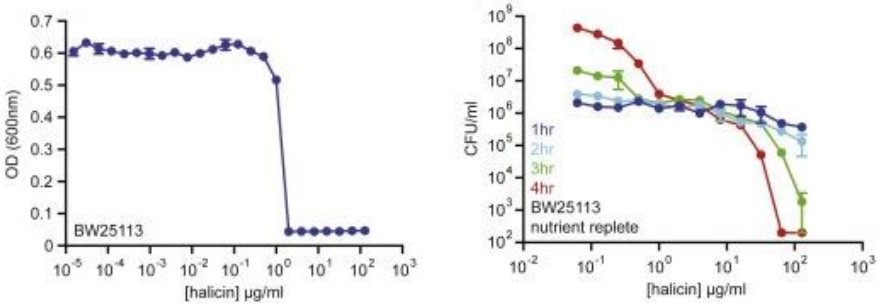
Results

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

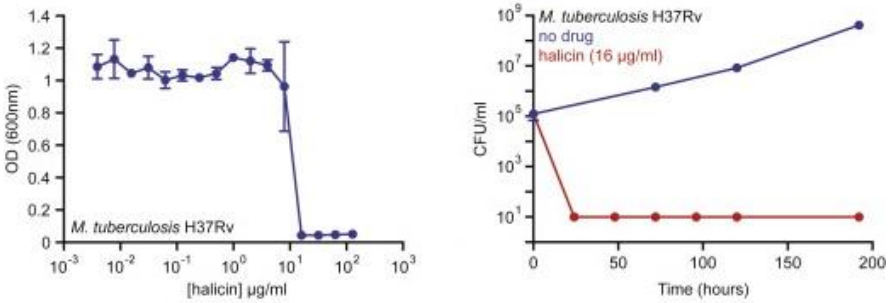
Halicin predicted to be antibacterial



Halicin against *E. coli*



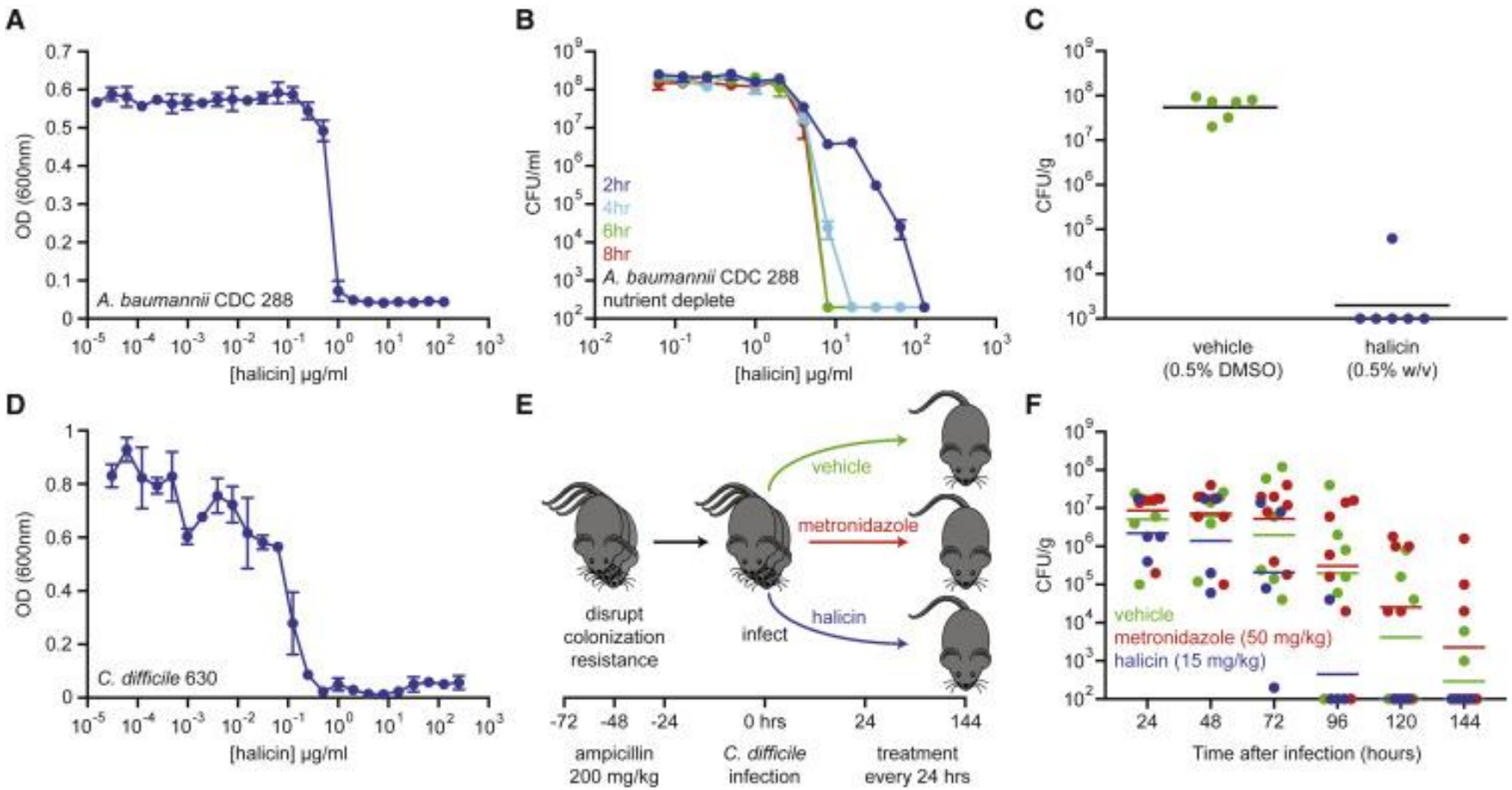
Halicin against *M. tuberculosis*



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

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

Application in medicine, patients-like-me retrieval

Finding patients with similar genetic and phenotypic features

Diagnostic odysseys

- Over 7,000 rare diseases, each affects < 200,000 patients in the US
  - Most diseases are phenotypically heterogeneous
  - Front-line clinicians might lack disease experience, resulting in expensive clinical workups for patients across multiple years
  - Diagnosis often requires a specialist, sub-specialist, or multi-disciplinary referrals
- On average, the long search for a **rare disease diagnosis takes 5 to 7 years, 4 up to 8 physicians, and 2 to 3 misdiagnoses**
- Diagnostic delay is so pervasive that it leads to problems for patients:
  - Undergoing **redundant testing and procedures**
  - Substantial delay in obtaining disease-appropriate management and **inappropriate therapies**
  - **Irreversible disease progression**—time window for intervention can be missed leading to disease progression

Can AI help shorten diagnostic odysseys for rare disease patients?

Deep learning for diagnosing patients with rare genetic diseases, *medRxiv*, 2022

AI-assisted medical diagnosis

- Deep learning models trained (via supervised learning) on large labeled datasets can achieve **near-expert clinical accuracy for common diseases**
- Existing models require **labeled datasets with thousands of diagnosed patients per disease**:
  - Diabetic retinopathy: deep neural net on 128 K retinal images
  - Skin lesions: deep neural net on 129 K clinical images of skin cancers
  - Childhood diseases: deep neural net on 1 M pediatric patient visits

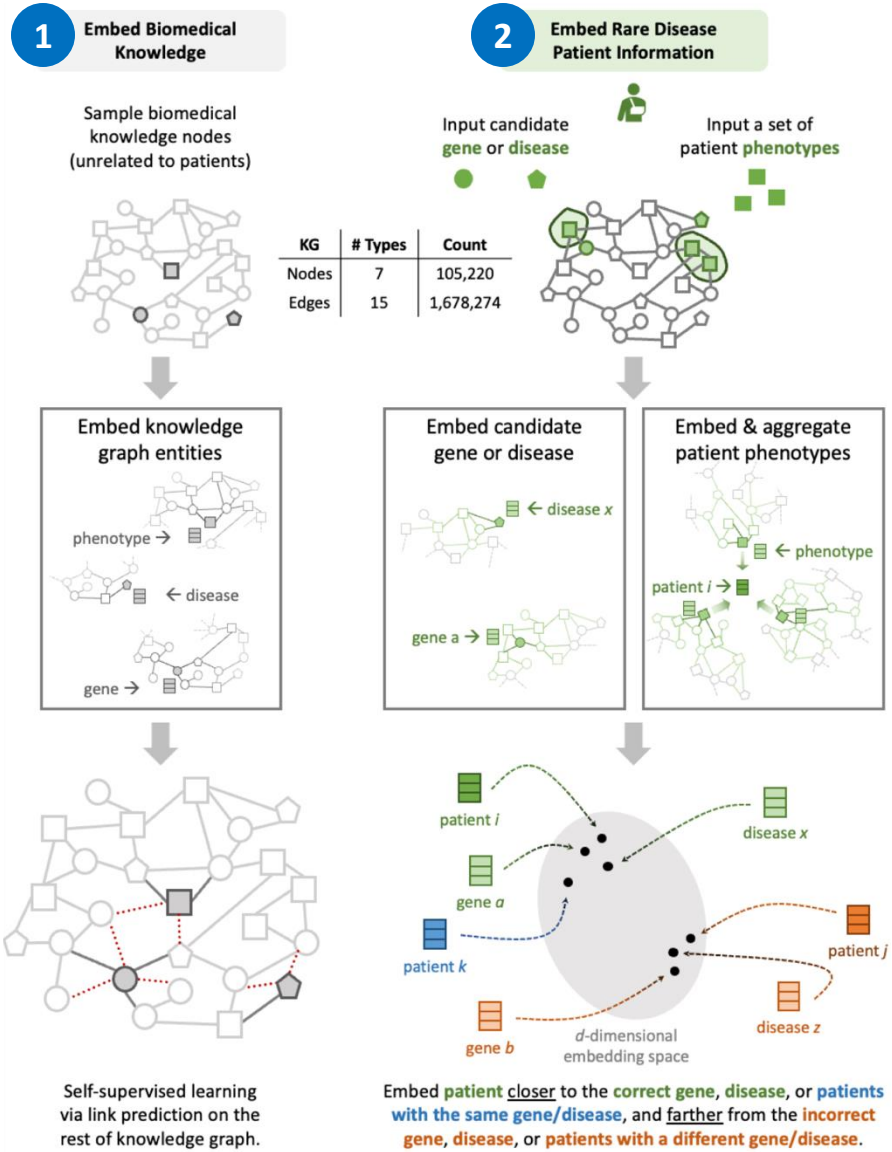
The challenge with rare diseases is fundamental — **datasets are three orders of magnitude smaller than in other uses of AI for medical diagnosis**

Needed is an entirely new approach to making AI-based rare disease diagnosis possible. This is for two primary reasons:

- Rare disease diagnosis cannot simply be solved by recruiting/labeling more patients because of high disease heterogeneity and low disease prevalence
- Rare disease diagnosis cannot be solved by supervised deep learning because the models cannot extrapolate to novel genetic diseases and atypical disease presentations

Deep learning for diagnosing patients with rare genetic diseases, *medRxiv,* 2022

Graph learning approach

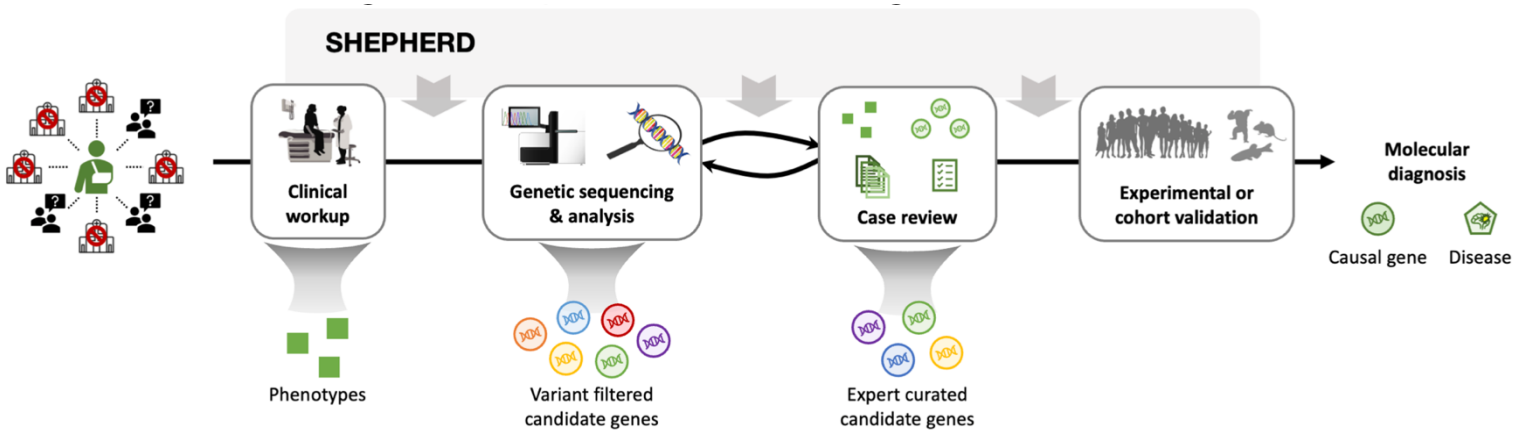


- Step 1: Incorporate knowledge of known phenotype, gene, and disease relationships via GNN
  - Knowledge-guided learning is achieved by self-supervised pre-training on our precision-medicine knowledge graph
- Step 2: Pre-trained GNN from Step 1 is fine-tuned using synthetic patients
  - Training exclusively on synthetic rare disease patients without the use of any real-world labeled cases
  - Synthetic patients used for training are created using an adaptive simulation approach
  - Realistic rare disease patients with varying numbers of phenotypes and candidate genes

Deep learning for diagnosing patients with rare genetic diseases, *medRxiv*, 2022

Diagnostic tasks

- Three diagnostic tasks:
  - **Causal gene discovery:** Given a patient's set of phenotypes and a list of genes in which the patient has mutations, **prioritize genes** harboring mutations that cause the disease (phenotypes)
  - **Patients-like-me:** Given a patient, **find other patients** with similar genetic and phenotypic features suitable for clinical follow-up
  - **Characterization of novel diseases:** Given a patient's phenotypes, **provide an interpretable NLP name** for the patient's disease based on its similarity to each disease in the KG



Deep learning for diagnosing patients with rare genetic diseases, *medRxiv*, 2022

Experimental setup

https://undiagnosed.hms.harvard.edu

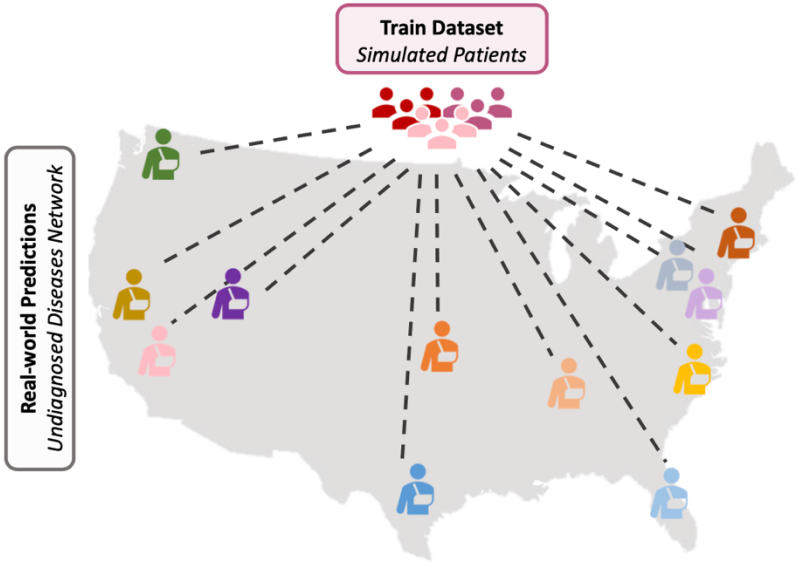
Undiagnosed Diseases Network

SHEPHERD's model training:

- 36K synthetic patients

SHEPHERD's model evaluation

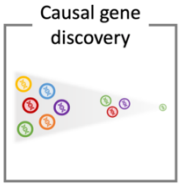
- **UDN patient cohort:** 465 rare disease patients with labeled diagnoses, spanning 299 diseases
  - 79% of genes and 83% of diseases are represented in only a single patient
- **MyGene2 patient cohort:** 146 rare disease patients, spanning 55 diseases

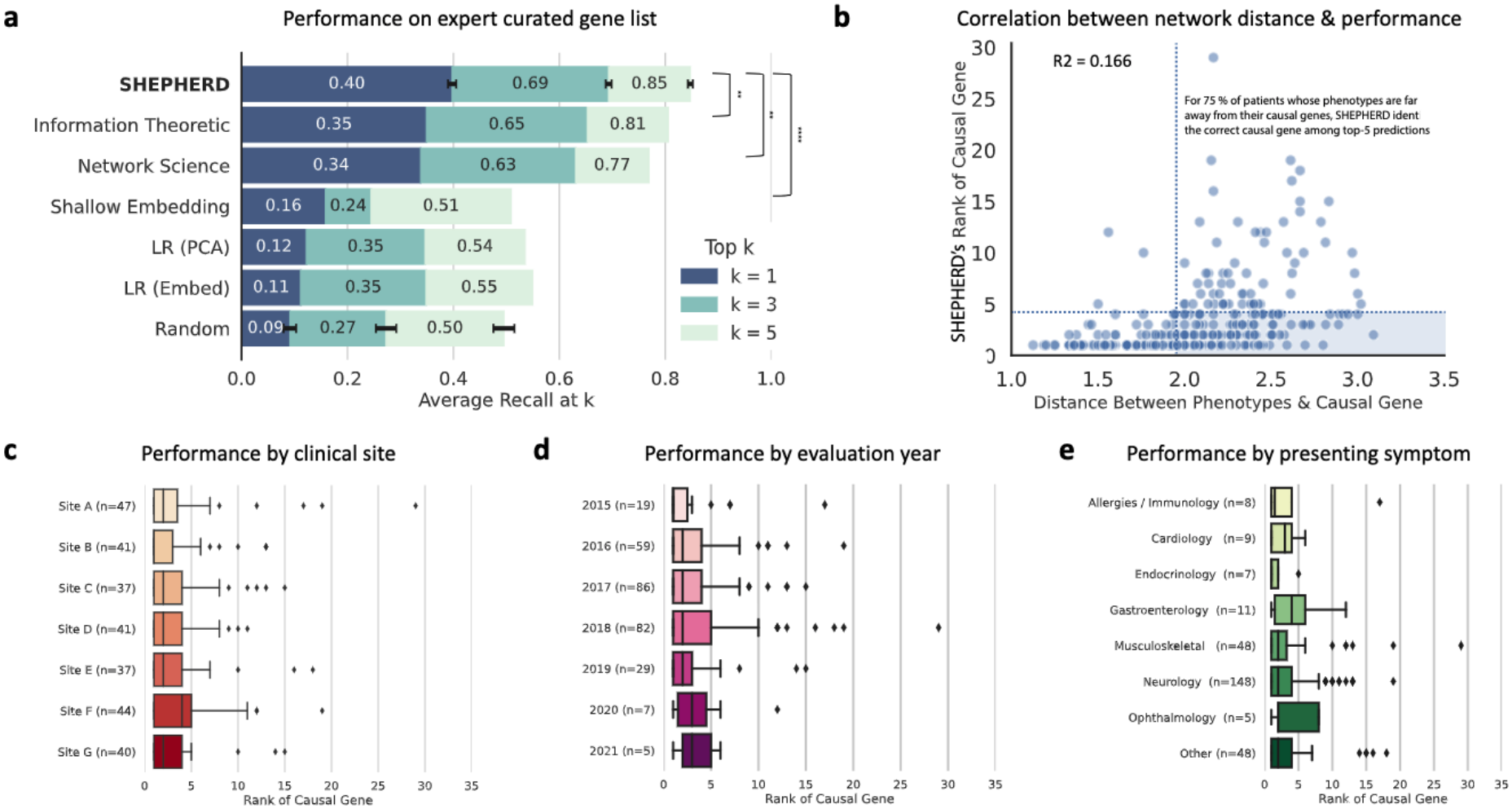


Patient dataset	Train cohort	Validation cohort	Test cohort
Simulated	N = 36,224	N = 6,400	---
UDN	---	---	N = 465
MyGene2	---	---	N = 146

Simulation of undiagnosed patients with novel genetic conditions, *medRxiv* 2022
Deep learning for diagnosing patients with rare genetic diseases, *medRxiv* 2022

Results: Disease gene discovery





Results: Patients-like-me

UMAP plot of SHEPHERD's embedding space of all simulated (circle), UDN (up-facing triangle), and MyGene2 (down-facing triangle) patients colored by their Orphanet disease category

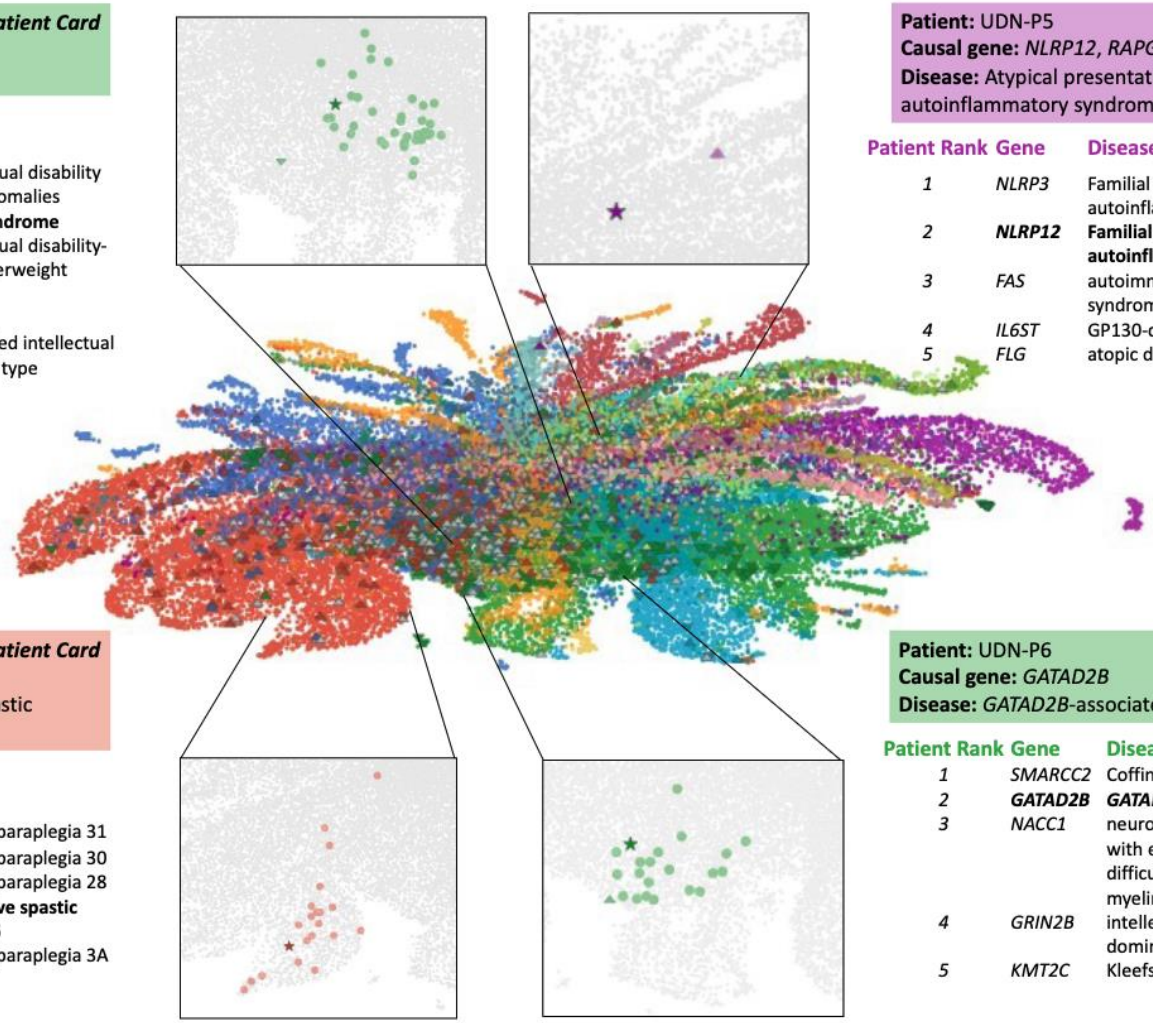
a

Patient: UDN-P3 ***Patient Card***
Causal gene: *RPS6KA3*
Disease: Coffin-Lowry syndrome

Patient Rank	Gene	Disease
1	*GRIA3*	X-linked intellectual disability due to GRIA3 anomalies
2	***RPS6KA3***	**Coffin-Lowry syndrome**
3	*THOC2*	X-linked intellectual disability-short stature-overweight syndrome
4	*AP1S2*	Fried syndrome
5	*SMS*	Syndromic X-linked intellectual disability Snyder type

Patient: UDN-P5 ***Patient Card***
Causal gene: *NLRP12, RAPGEFL1*
Disease: Atypical presentation of familial cold autoinflammatory syndrome

Patient Rank	Gene	Disease
1	*NLRP3*	Familial cold-induced autoinflammatory syndrome 1
2	***NLRP12***	**Familial cold-induced autoinflammatory syndrome 2**
3	*FAS*	autoimmune lymphoproliferative syndrome type 1
4	*IL6ST*	GP130-deficient hyper-IgE syndrome
5	*FLG*	atopic dermatitis 2



Patient: UDN-P4 ***Patient Card***
Causal gene: *CAPN1*
Disease: autosomal recessive spastic paraplegia type 76

Patient Rank	Gene	Disease
1	*REEP1*	hereditary spastic paraplegia 31
2	*KIF1A*	hereditary spastic paraplegia 30
3	*DDHD1*	hereditary spastic paraplegia 28
4	***CAPN1***	**autosomal recessive spastic paraplegia type 76**
5	*MTPAP*	hereditary spastic paraplegia 3A

Patient: UDN-P6 ***Patient Card***
Causal gene: *GATAD2B*
Disease: *GATAD2B*-associated syndrome

Patient Rank	Gene	Disease
1	*SMARCC2*	Coffin-Siris syndrome 8
2	***GATAD2B***	***GATAD2B*-associated syndrome**
3	*NACC1*	neurodevelopmental disorder with epilepsy, cataracts, feeding difficulties, and delayed brain myelination syndrome
4	*GRIN2B*	intellectual disability, autosomal dominant 6
5	*KMT2C*	Kleefstra syndrome

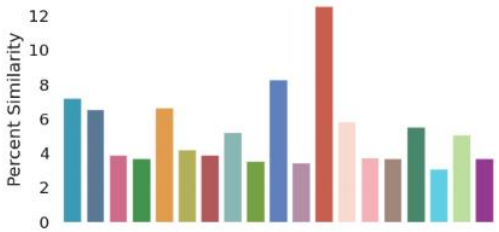
Deep learning for diagnosing patients with rare genetic diseases, *medRxiv*, 2022

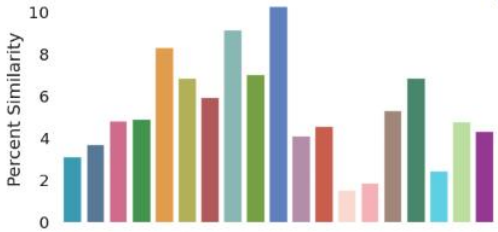
Results: New disease naming

Novel disease characterization

a

Rank	Disease
1	AR limb-girdle muscular dystrophy type 2B
2	GNE myopathy
3	MYH7-related late-onset scapuloperoneal muscular dystrophy
4	Emery-Dreifuss muscular dystrophy 2, AD
5	AR limb-girdle muscular dystrophy type 2G

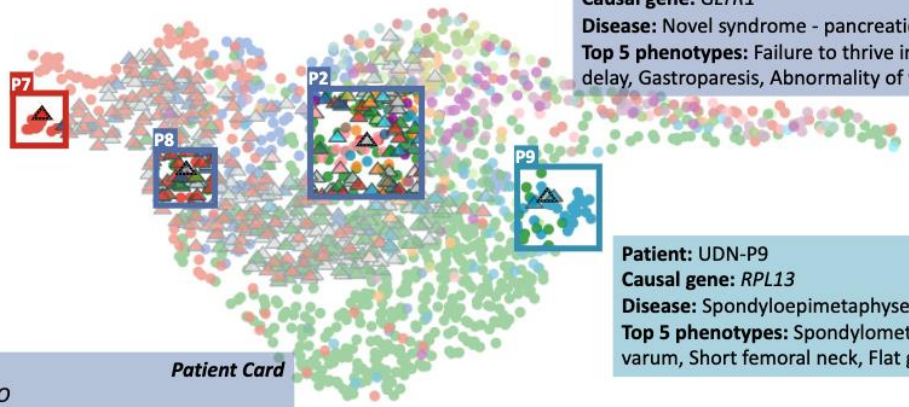




Rank	Disease
1	Methylmalonic aciduria & homocystinuria type cblF
2	Neonatal hemochromatosis
3	Homozygous 11P15-p14 deletion syndrome
4	ALG8-CDG
5	Congenital anemia

Patient: UDN-P2 ***Patient Card***
Causal gene: *GLYR1*
Disease: Novel syndrome - pancreatic insufficiency & malabsorption
Top 5 phenotypes: Failure to thrive in infancy, Global developmental delay, Gastroparesis, Abnormality of vision, Duodenal atresia

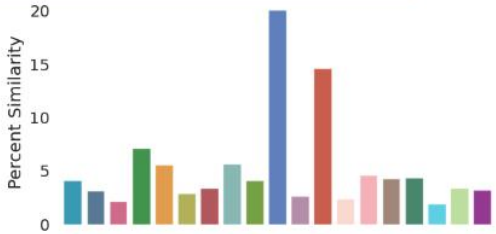
Patient: UDN-P7 ***Patient Card***
Causal gene: *SGCA*
Disease: AR limb-girdle muscular atrophy type 2D
Top 5 phenotypes: Toe walking, Calf muscle pseudohypertrophy, Elevated serum creatine kinase, Proximal muscle weakness, Generalized muscle weakness

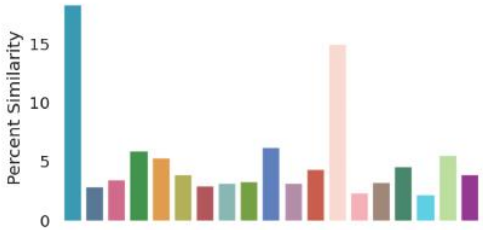


Patient: UDN-P9 ***Patient Card***
Causal gene: *RPL13*
Disease: Spondyloepimetaphyseal dysplasia, Isidor-Toutain type
Top 5 phenotypes: Spondylometaphyseal dysplasia, Genu varum, Short femoral neck, Flat glenoid fossa, Platyspondyly

Patient: UDN-P8 ***Patient Card***
Causal gene: *ATP5PO*
Disease: *ATP5PO*-related Leigh syndrome
Top 5 phenotypes: Profound global developmental delay, cerebral hypomyelination, limb hypertonia, hypoplasia of the corpus callosum, infantile spasms

Rank	Disease
1	Combined oxidative phosphorylation deficiency 39
2	Hypomyelinating leukodystropy-20
3	Pyruvate dehydrogenase E3-binding protein deficiency
4	Intellectual disability-epilepsy-extrapyramidal syndrome
5	Combined oxidative phosphorylation defect type 27





Rank	Disease
1	Multiple epiphyseal dysplasia type 1
2	Progressive pseudorheumatoid arthropathy of childhood
3	Multiple epiphyseal dysplasia type 5
4	Metaphyseal chondrodysplasia, Spahr type
5	Multiple epiphyseal dysplasia

Take-away messages

- SHEPHERD overcomes limitations of standard machine learning:
  - Model inputs as **KG subgraphs** (i.e., clinic-genetic subgraphs of patients)
  - Use **self-supervised pre-training on biomedical knowledge**
  - Train the model on a large cohort of **synthetic patients**
- SHEPHERD generalizes to novel phenotypes, genes, and diseases:
  - Performs well on patients whose **subgraphs are of varying size**
  - Performs well on **diagnosing patients with novel diseases**
- Implications:
  - Implications for **generalist models applicable across diagnostic process**
  - New opportunities to **shorten the diagnostic odyssey for rare disease**
  - Implications for using **deep learning on medical datasets with very few labels**

First deep learning approach for individualized diagnosis of rare genetic diseases

Graph learning approach is not only helpful but necessary