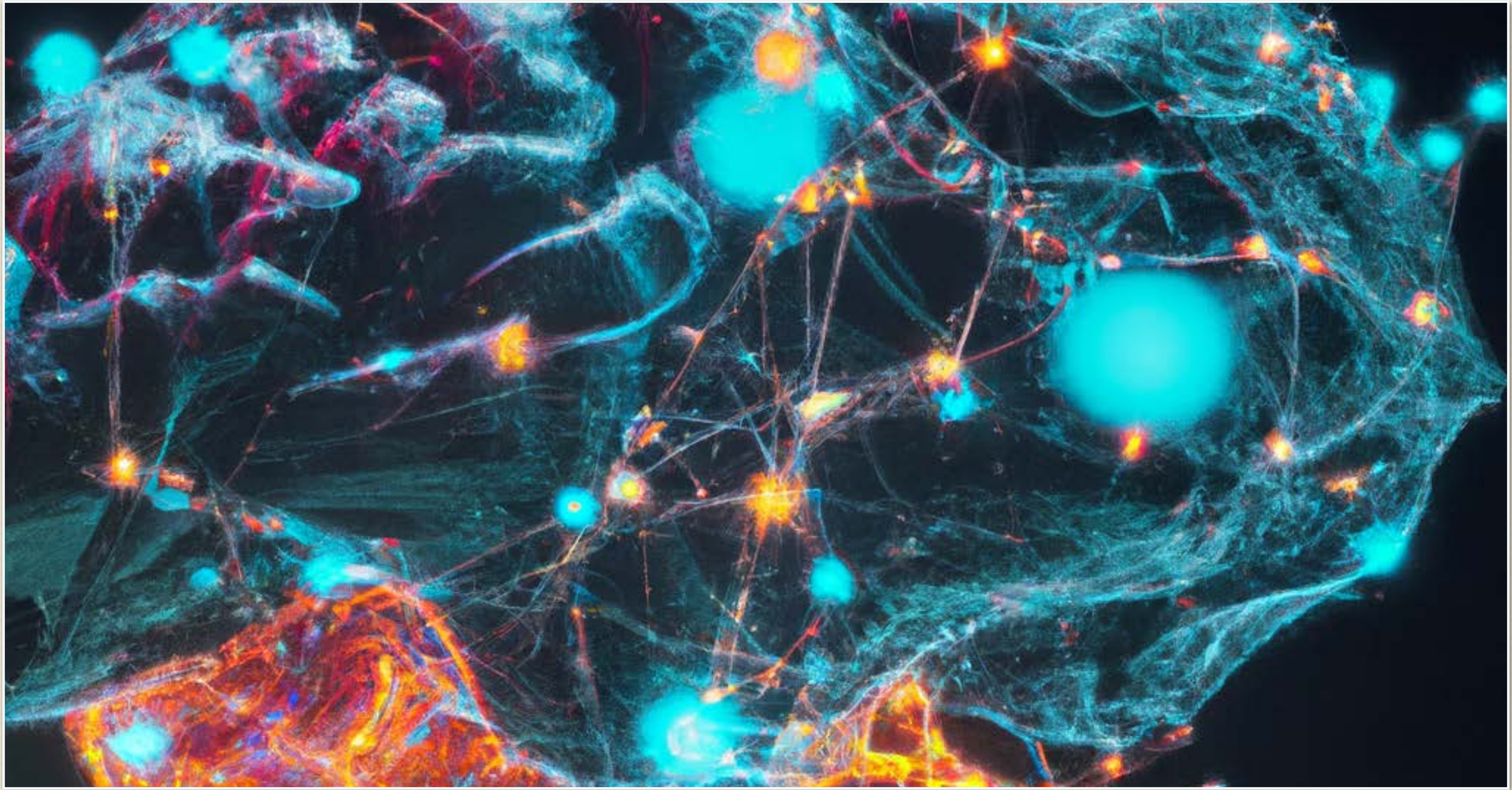
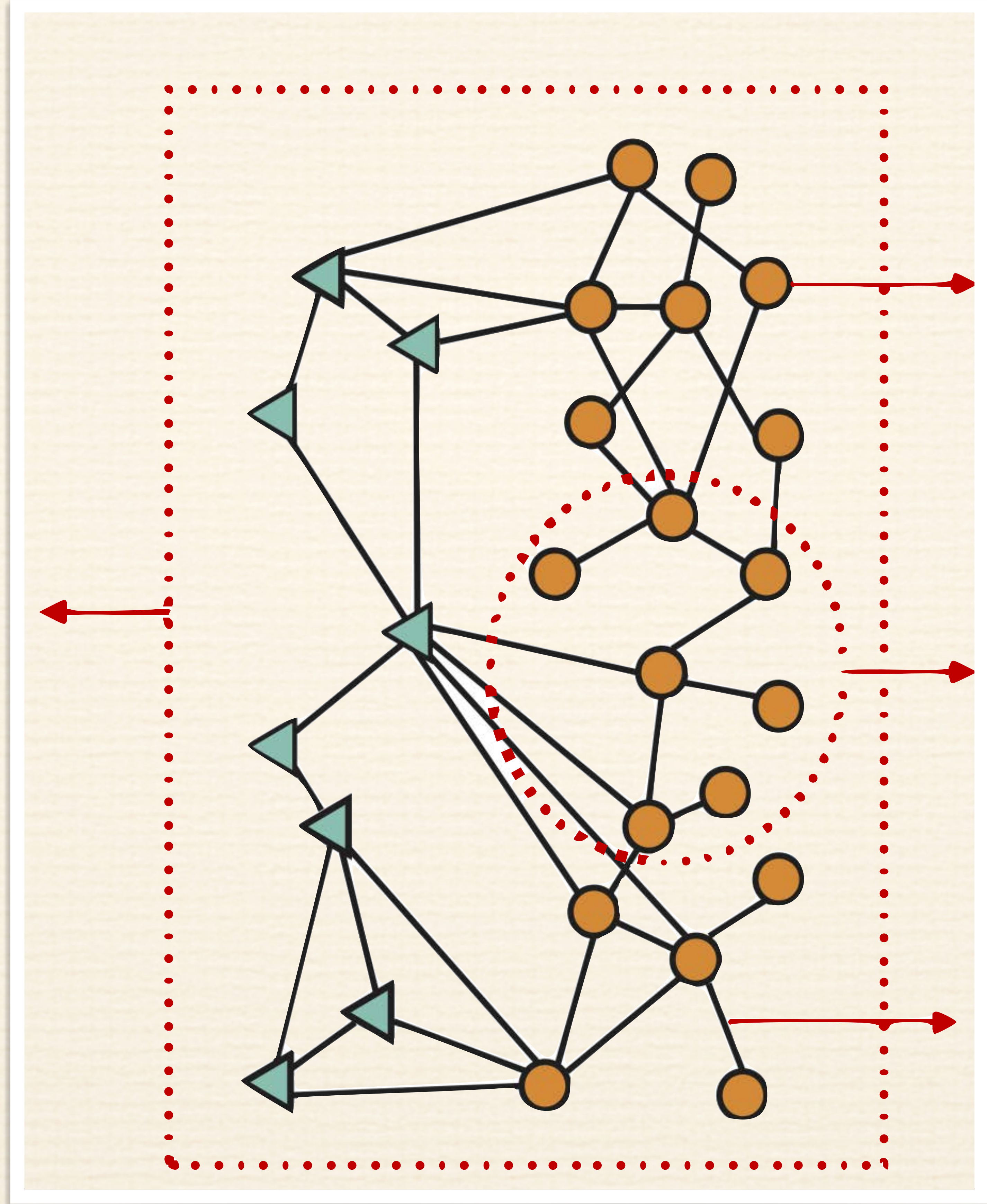


The Power of Graph Learning

Floris Geerts (University of Antwerp, Belgium)

Graph learning

Prediction and classification problems on graphs



Vertex level

Graph level

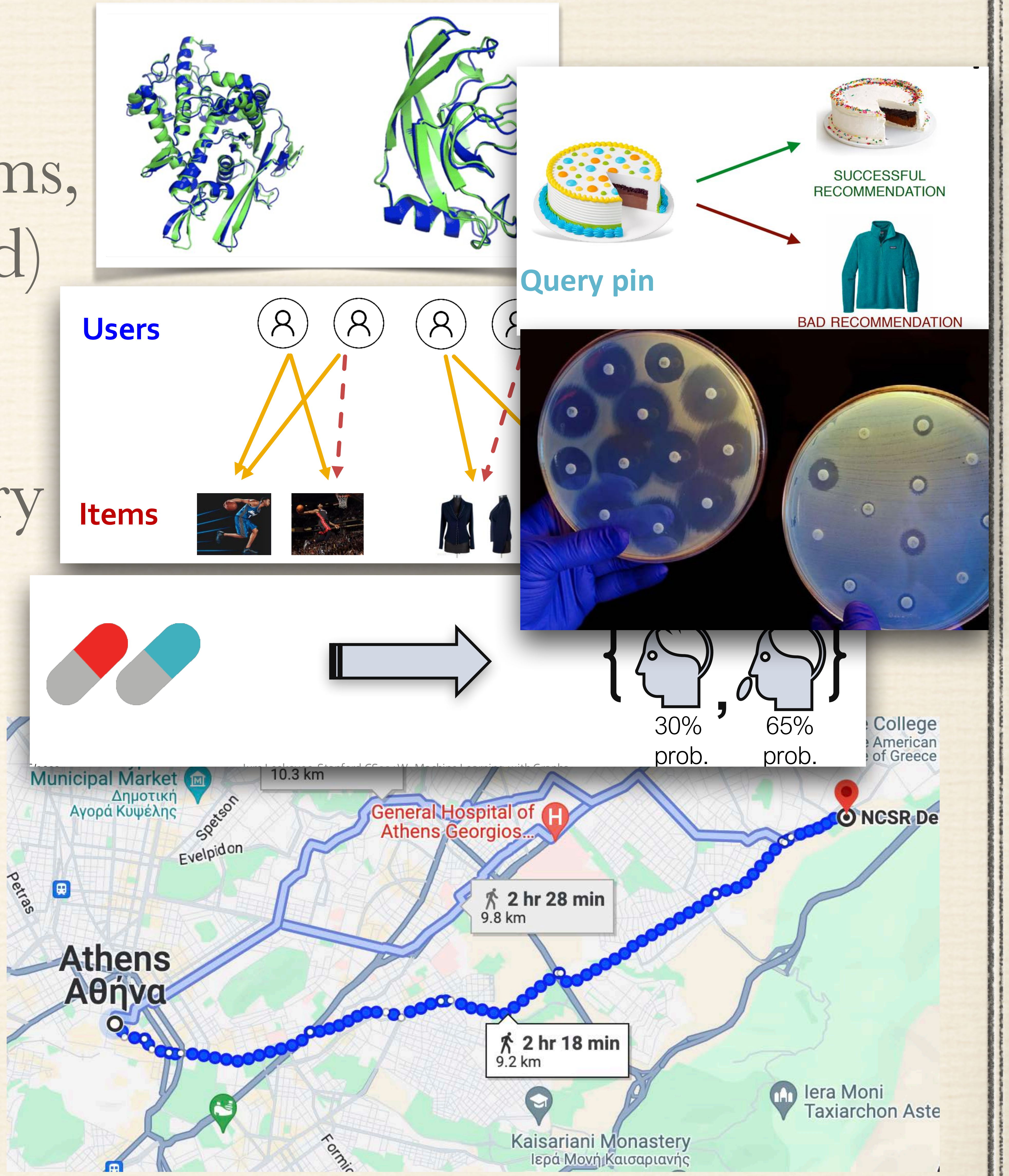
Subgraph level

Edge/link level

Image: Machine Learning on Graphs, Stanford course Jure Leskovec

Examples

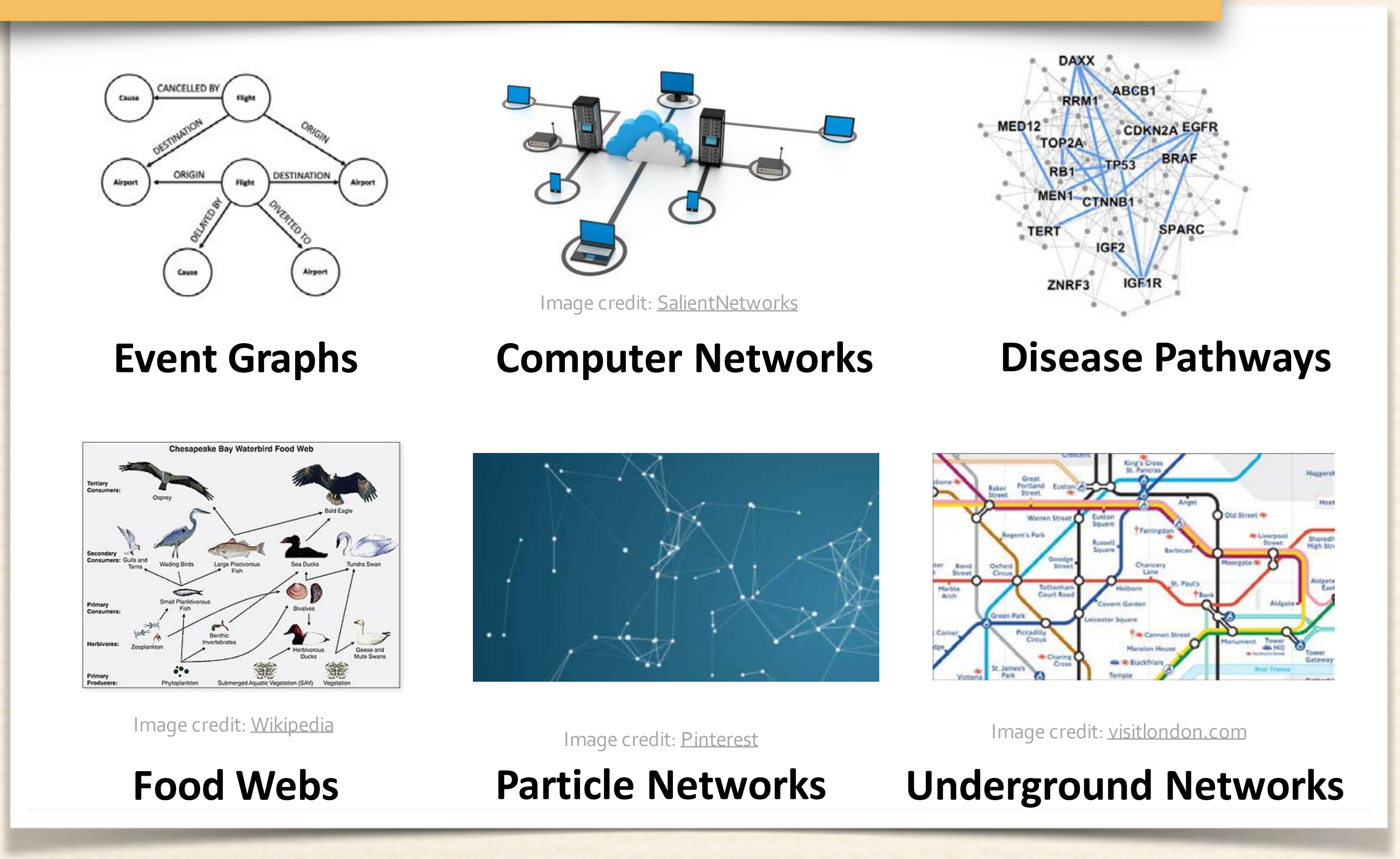
- Vertex classification: categorise online user/items, location amino acids (protein folding, alpha fold)
- Link prediction: knowledge graph completion, recommender systems, drug side effect discovery
- Graph classification: molecule property, drug discovery
- Subgraph tasks: traffic prediction

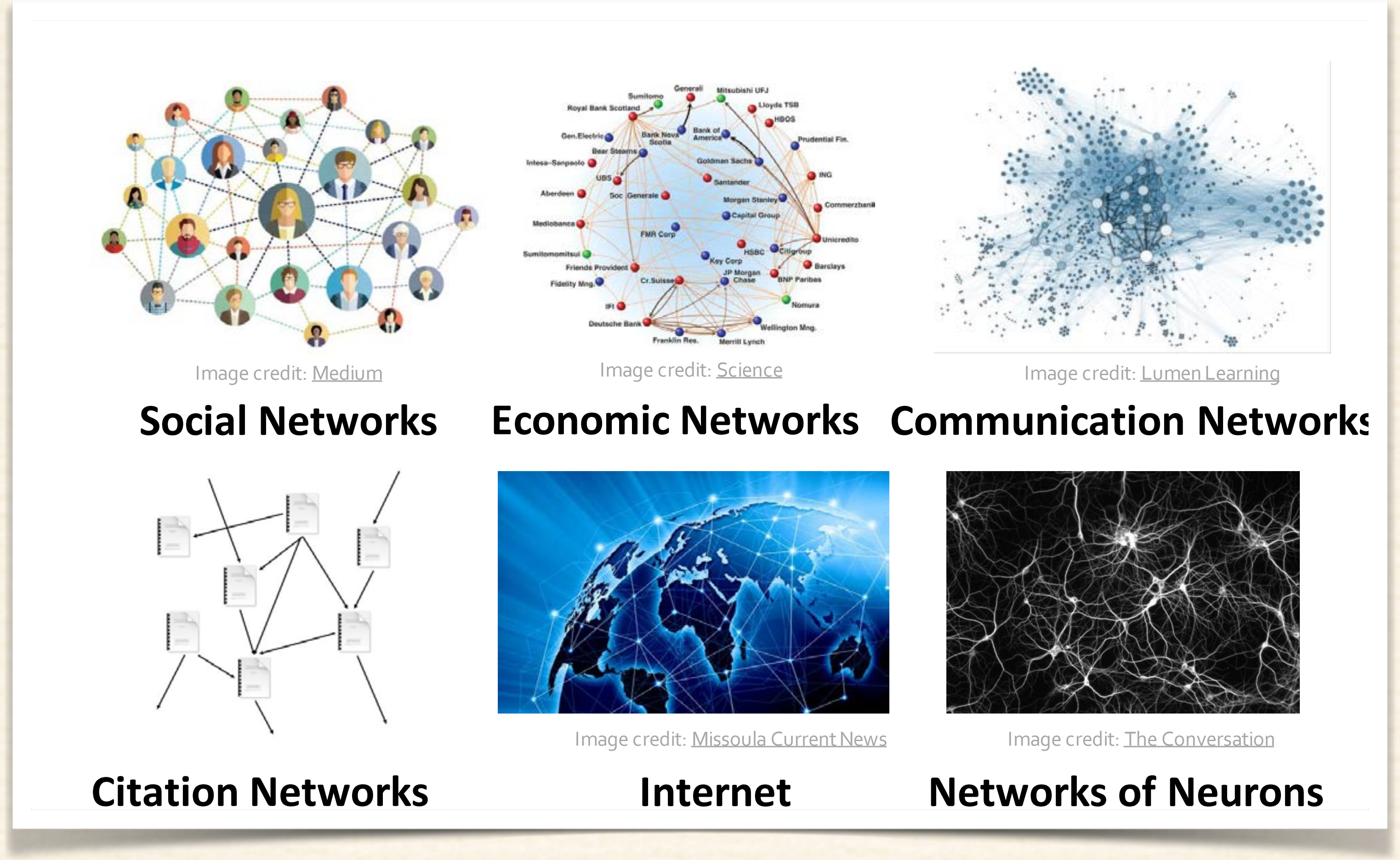


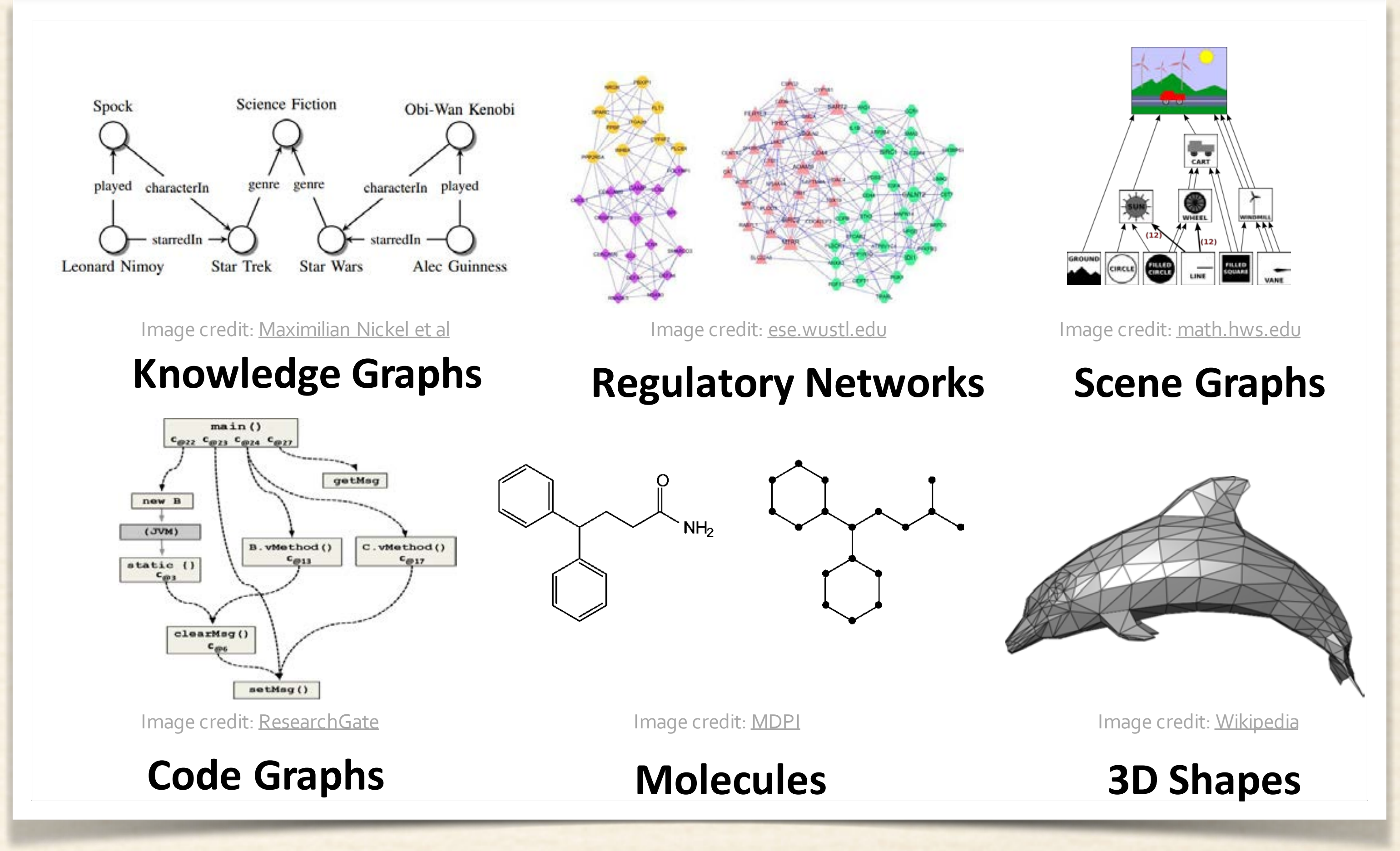
Images: Machine Learning on Graphs, Stanford course Jure Leskovec

Why learning on graphs?

Graphs are everywhere!





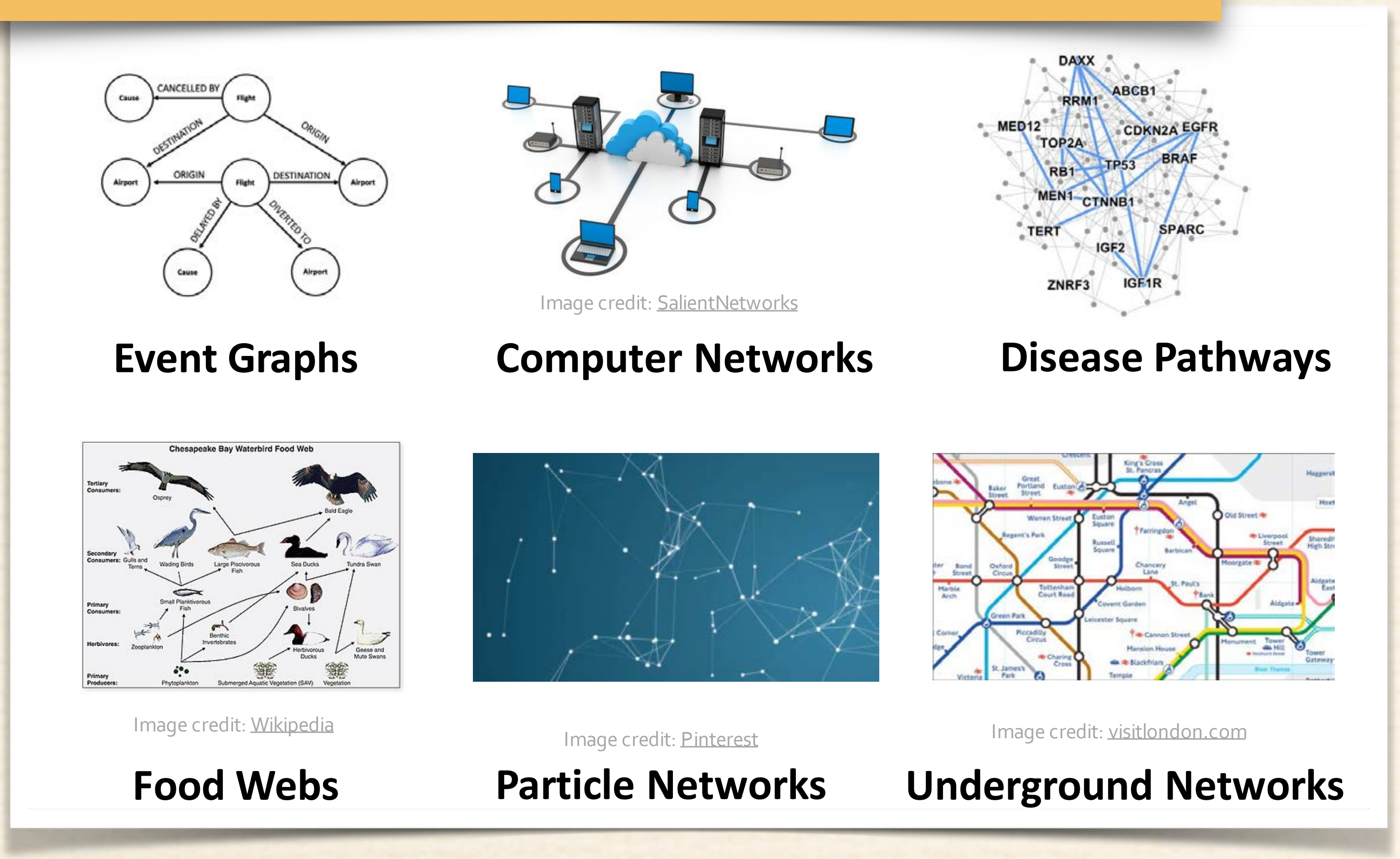


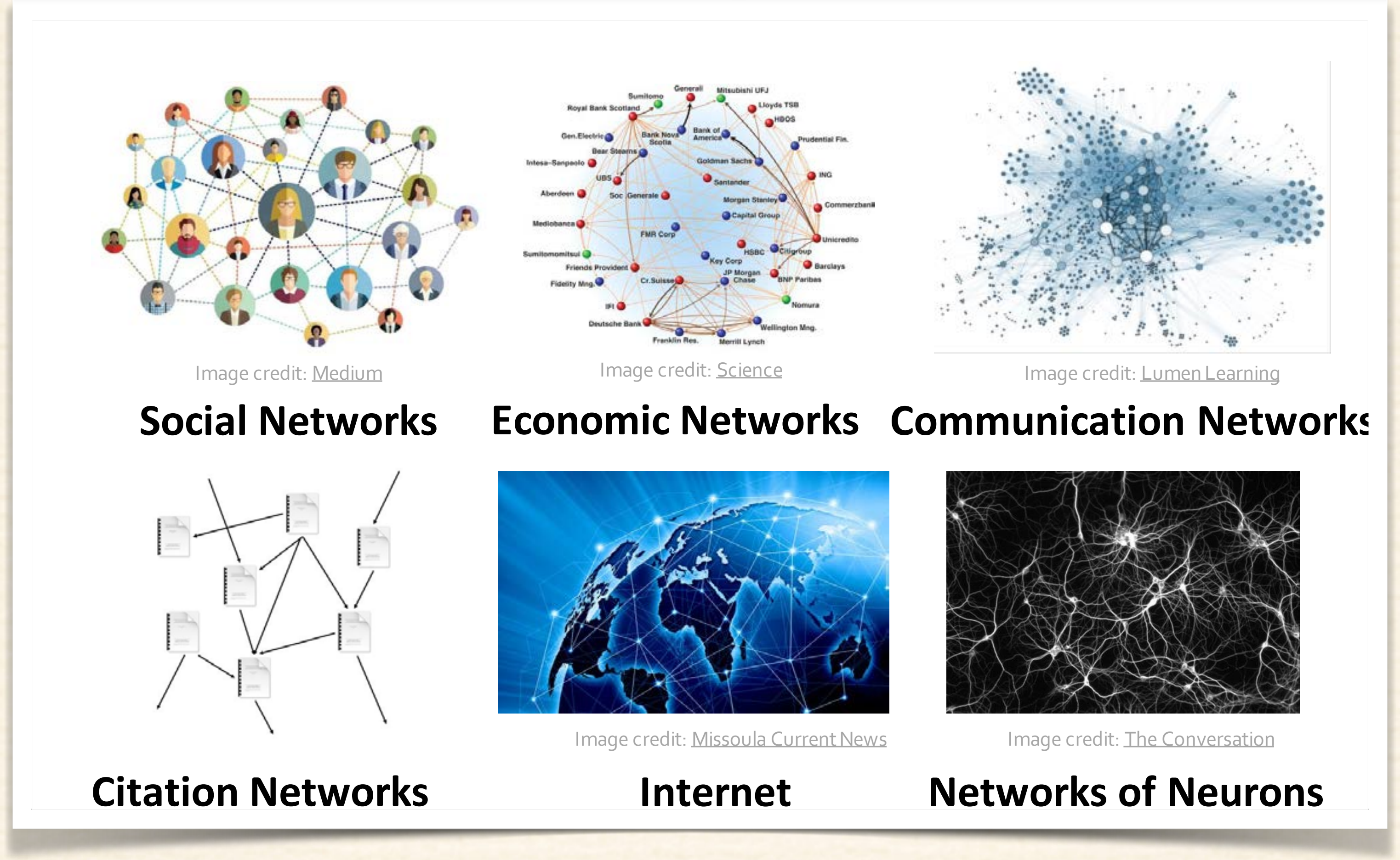
Graph learning methods are thus widely applicable

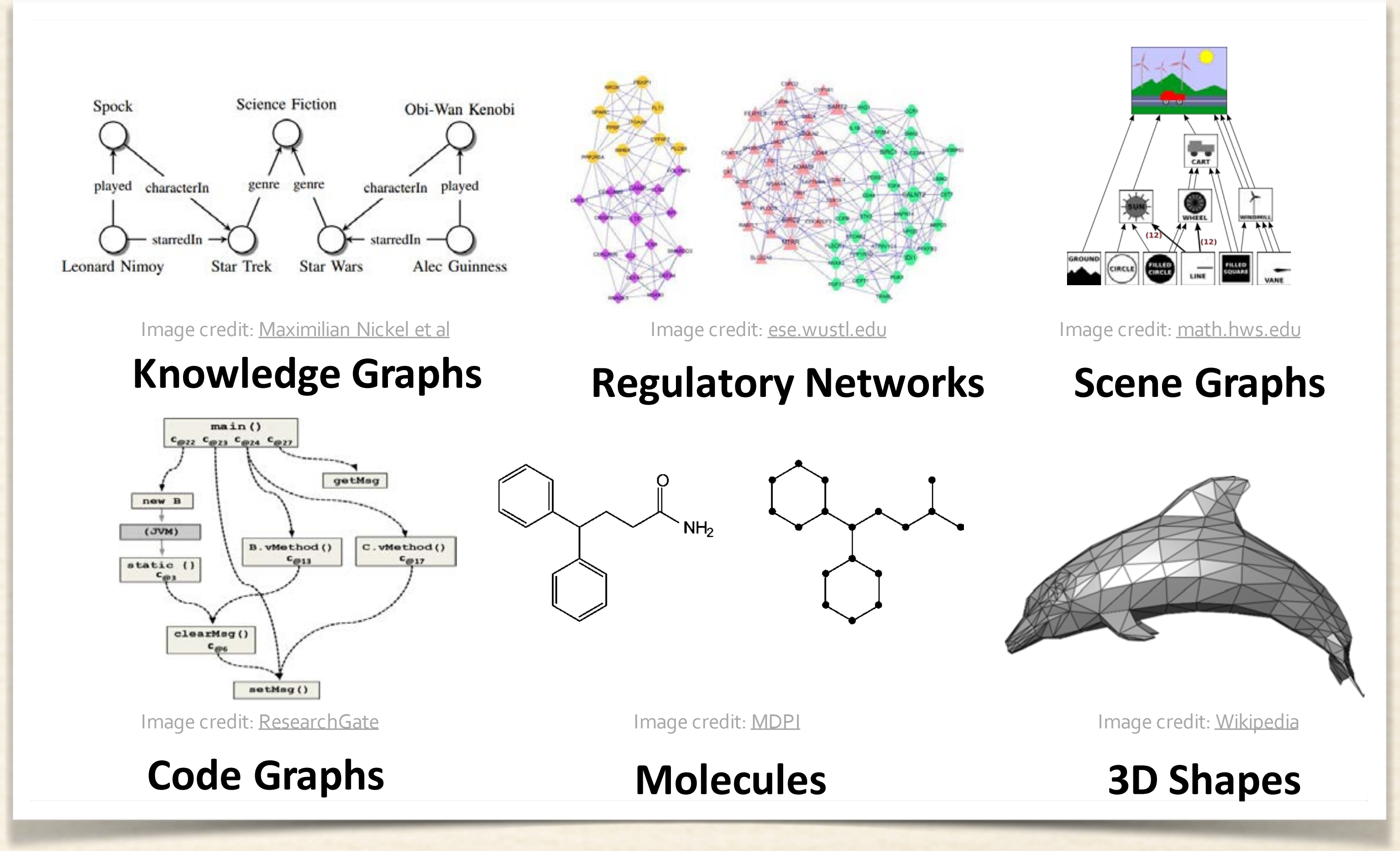
Images: Machine Learning on Graphs, Stanford course Jure Leskovec

Why learning on graphs?

Graphs are everywhere!





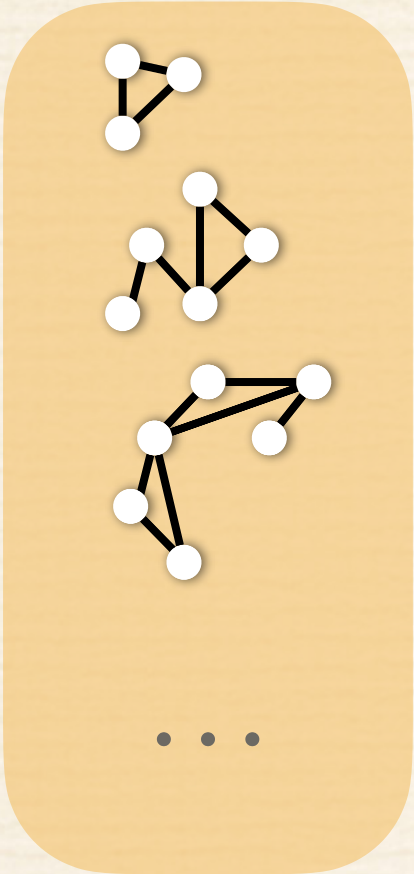


Graph learning methods are thus widely applicable

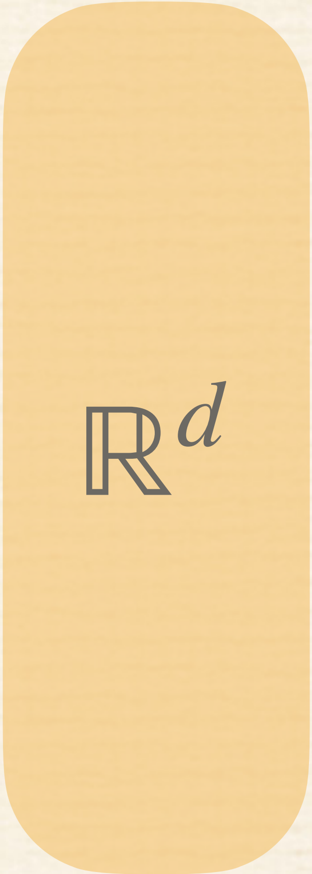
How is learning typical done?

Images: Machine Learning on Graphs, Stanford course Jure Leskovec

Embedding-based graph learning

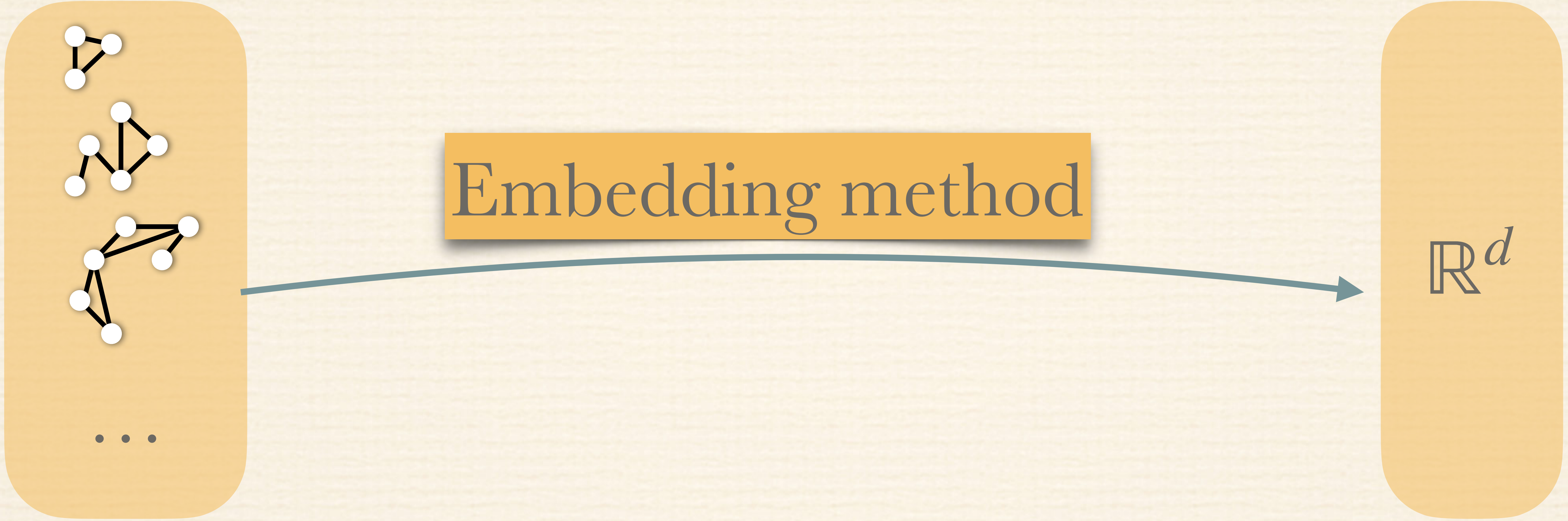


$\mathcal{G} =$ all graphs



$\mathbb{Y} =$ output space

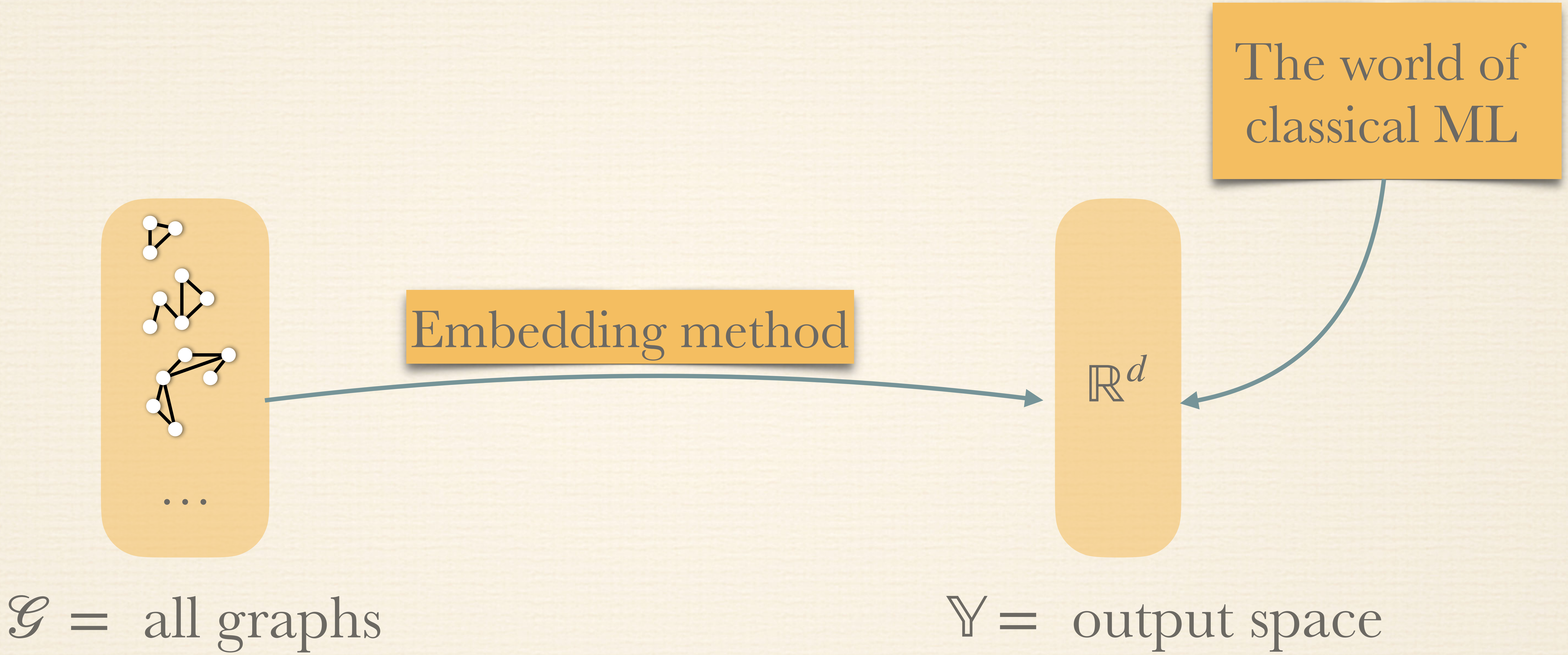
Embedding-based graph learning



$\mathscr{G} =$ all graphs

$\mathbb{Y} =$ output space

Embedding-based graph learning



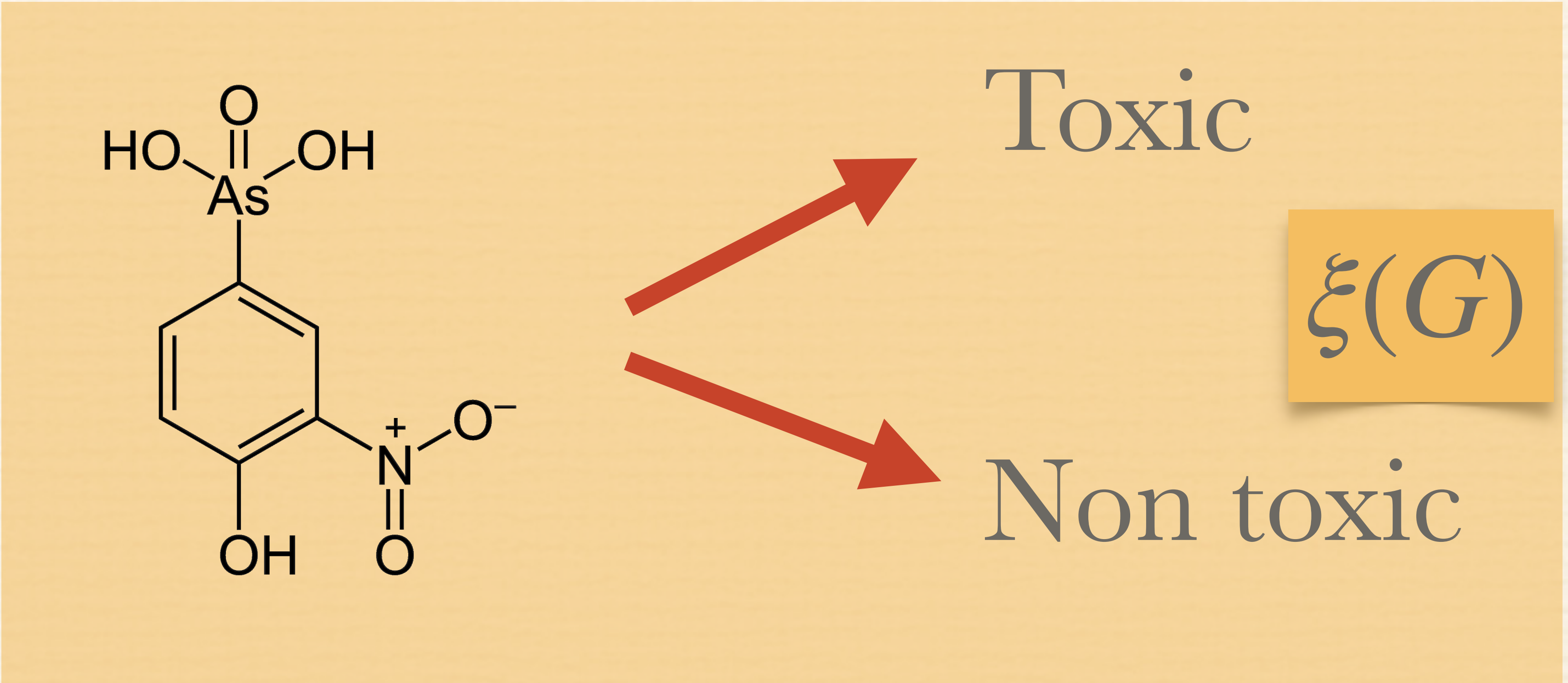
Embeddings

$\mathscr{G} =$ all graphs
$\mathscr{V} =$ all vertices
$\mathbb{Y} =$ output space

- Graph embedding: $\xi : \mathscr{G} \to \mathbb{Y}$
- Vertex embedding: $\xi : \mathscr{G} \to (\mathscr{V} \to \mathbb{Y})$
- p-Vertex embedding: $\xi : \mathscr{G} \to (\mathscr{V}^p \to \mathbb{Y})$

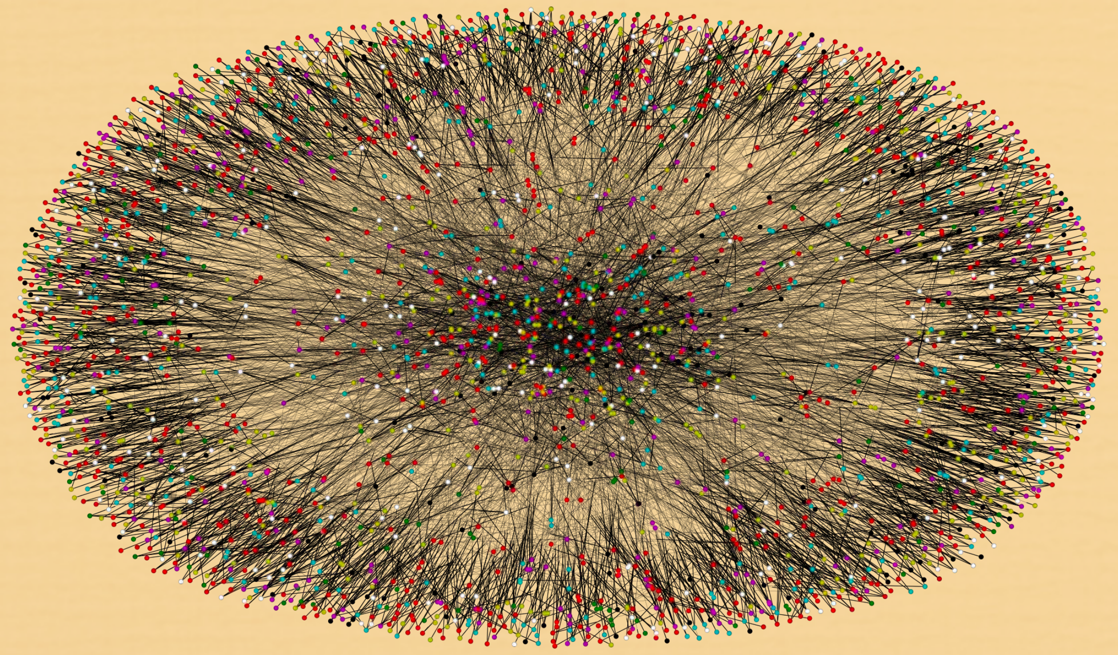
Graph embeddings

- Graph embedding: $\xi : \mathscr{G} \rightarrow \mathbb{Y}$
- Graph classification/regression



Vertex embeddings

- Vertex embedding: $\xi : \mathcal{G} \to (\mathcal{V} \to \mathbb{Y})$
- Vertex classification/regression. For example, prediction of subject of papers.



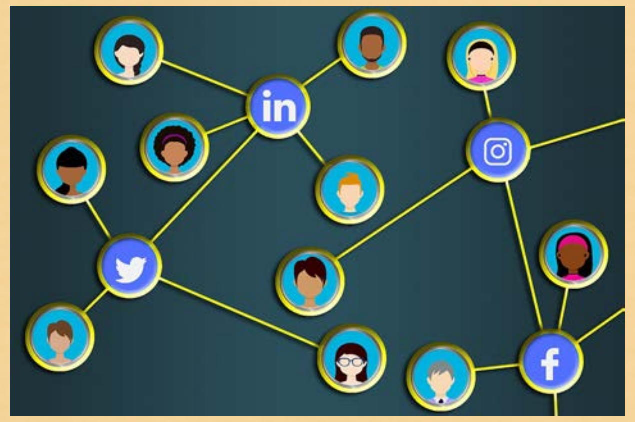
$$\to \quad \begin{array}{ccc} & & \xi(G, v) \\ \text{paper}_1 & \to & \text{math} \\ \text{paper}_2 & \to & \text{computer science} \\ \vdots & \vdots & \vdots \\ \text{paper}_n & \to & \text{biology} \end{array}$$

Images: Cora dataset

p-Vertex embeddings

- p-Vertex embedding: $\xi : \mathscr{G} \to (\mathscr{V}^p \to \mathbb{Y})$

- For example, 2-vertex embeddings: link prediction



$\to$ (Adam, Eve) $\quad \mapsto$ link

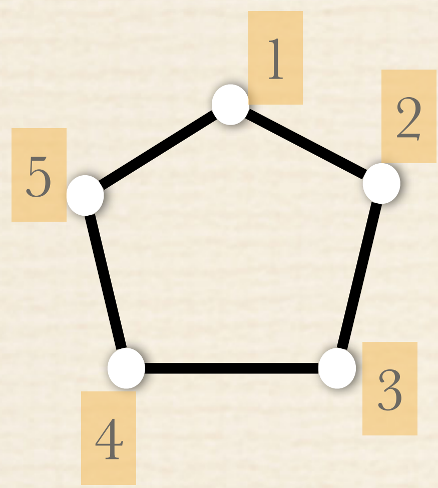
(Trump, Biden) $\quad \mapsto$ no link

$\dots$

$\xi(G, v, w)$

What makes graphs special?

$\mathcal{G}$



$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

flatten

$\mathbb{Y} = \mathbb{R}^{10}$

(0,1,0,0,1,1,0,1,0,0,0,1,0,1,0,0,0,1,0,1,1,0,0,1,0)

Deep neural network

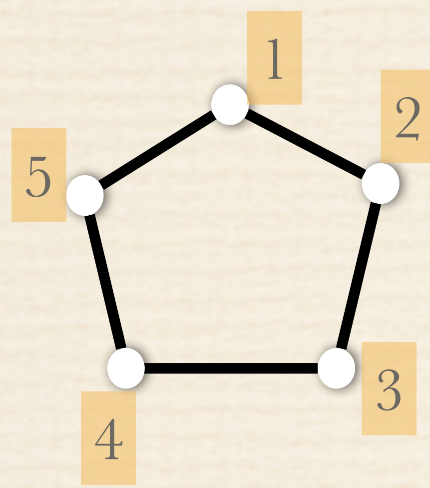
Support vector machines

…

Images: flaticon.com, Noun project

What makes graphs special?

$\mathcal{G}$

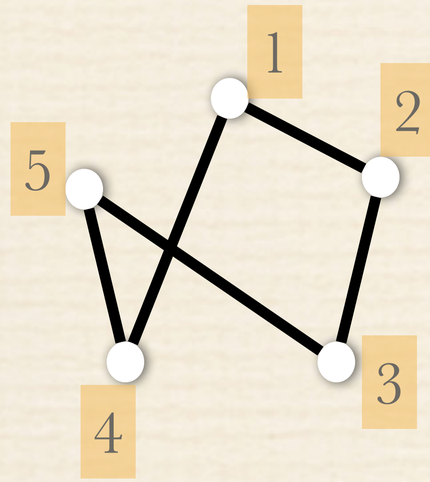


$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

flatten

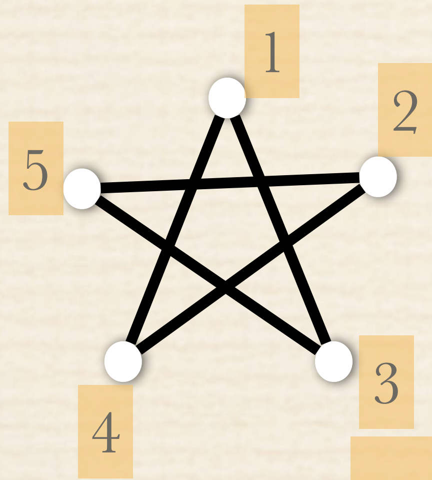
$\mathbb{Y} = \mathbb{R}^{10}$

$\longrightarrow$ (0,1,0,0,1,1,0,1,0,0,0,1,0,1,0,0,0,1,0,1,1,0,0,1,0) $\longrightarrow$



$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

$\longrightarrow$ (0,1,0,1,0,1,0,1,0,0,0,1,0,0,1,1,0,0,0,1,0,0,1,1,0) $\longrightarrow$



$$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$\longrightarrow$ (0,0,1,1,0,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,0,1,1,0,0) $\longrightarrow$

permuted adjacency matrices

Deep neural network

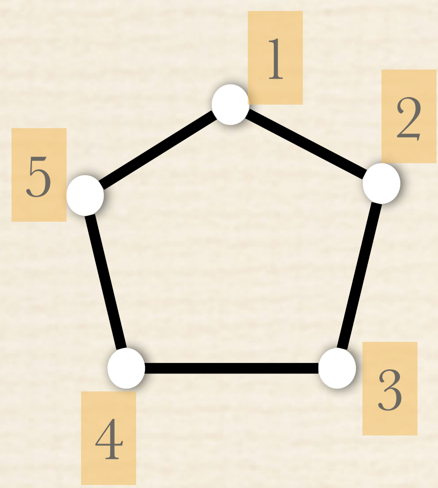
Support vector machines

…

Images: flaticon.com, Noun project

What makes graphs special?

$\mathcal{G}$

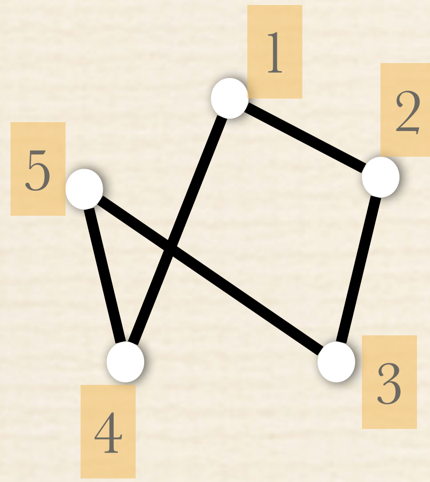


$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

flatten

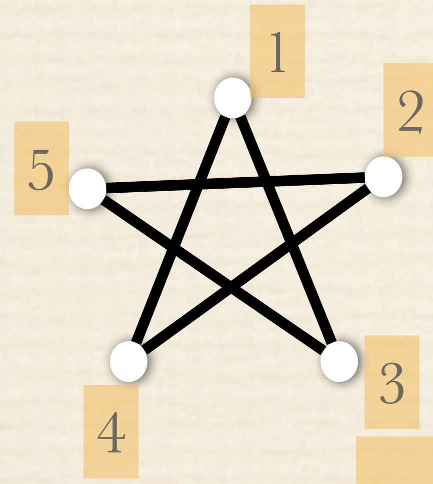
$\mathbb{Y} = \mathbb{R}^{10}$

$\longrightarrow$ (0,1,0,0,1,1,0,1,0,0,0,1,0,1,0,0,0,1,0,1,1,0,0,1,0) $\longrightarrow$



$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

$\longrightarrow$ (0,1,0,1,0,1,0,1,0,0,0,1,0,0,1,1,0,0,0,1,0,0,1,1,0) $\longrightarrow$



$$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$\longrightarrow$ (0,0,1,1,0,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,0,1,1,0,0) $\longrightarrow$

permuted adjacency matrices

Deep neural network

Support vector machines

…

Different representation ⇒ different result

Images: flaticon.com, Noun project

Invariant embeddings

- We need embeddings to be graph invariants
- Isomorphic inputs should give the same result

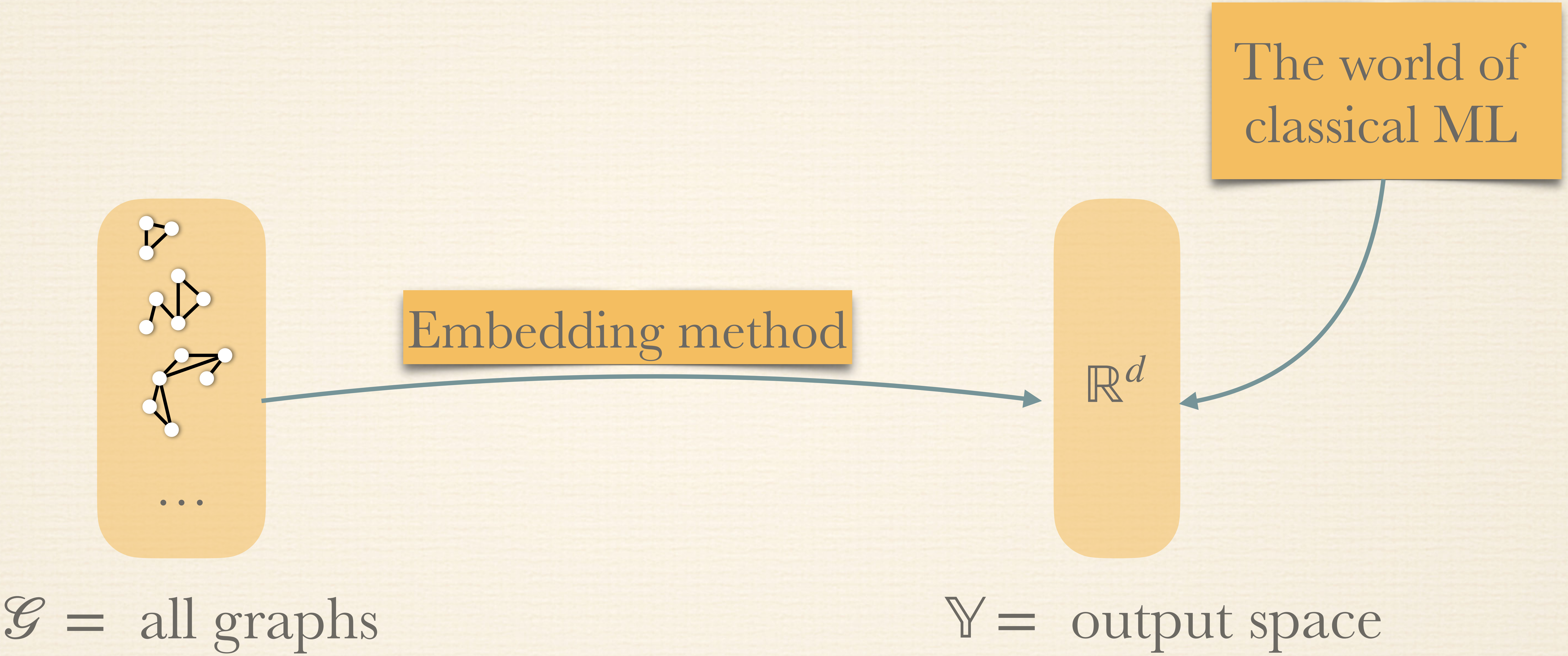
Invariant embeddings

- We need embeddings to be graph invariants
- Isomorphic inputs should give the same result

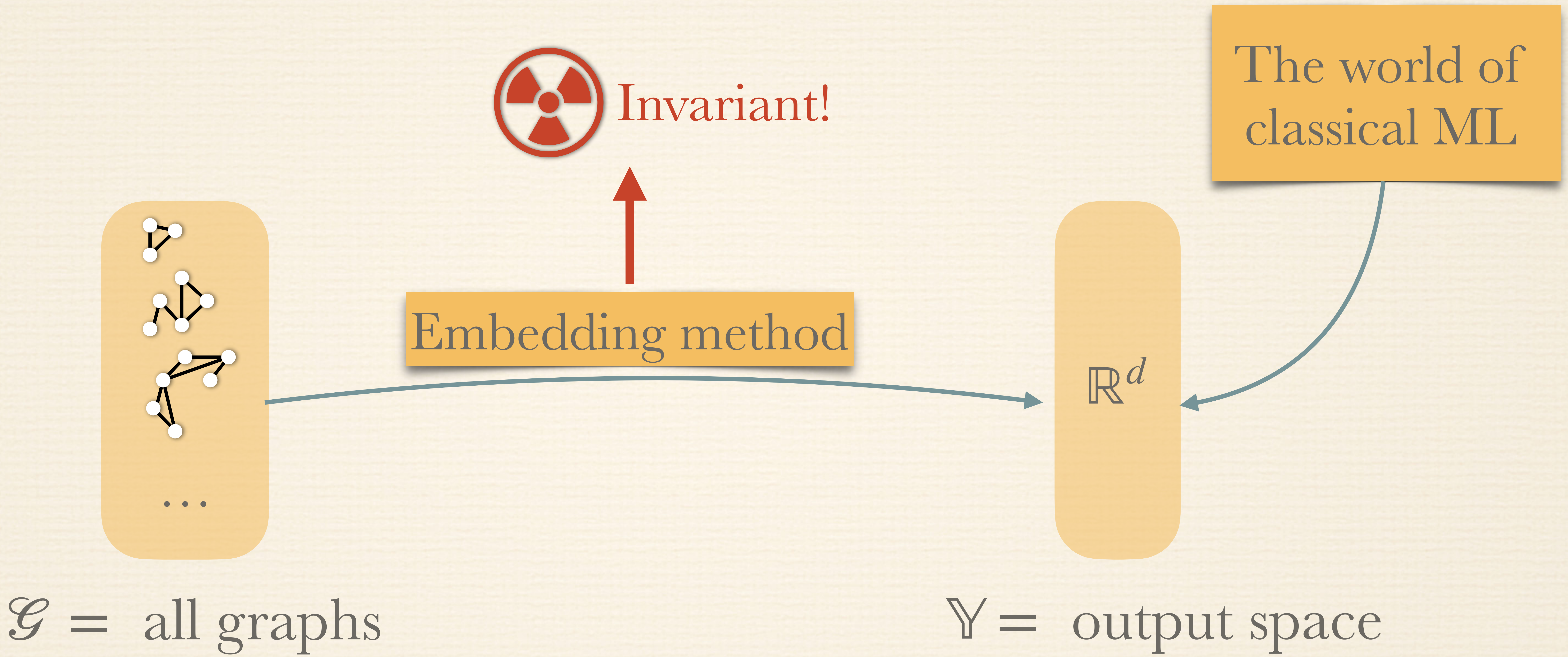
p-vertex embedding $\xi : \mathscr{G} \to (\mathscr{V}^p \to \mathbb{Y}) : (G, \mathbf{v}) \mapsto \xi(G, \mathbf{v})$ is invariant if for all G, all isomorphisms π, and $\mathbf{v} \in V_G^p$: $\xi(G, \mathbf{v}) = \xi(\pi(G), \pi(\mathbf{v}))$

- Invariance is typically achieved by composing invariant building blocks to build embeddings

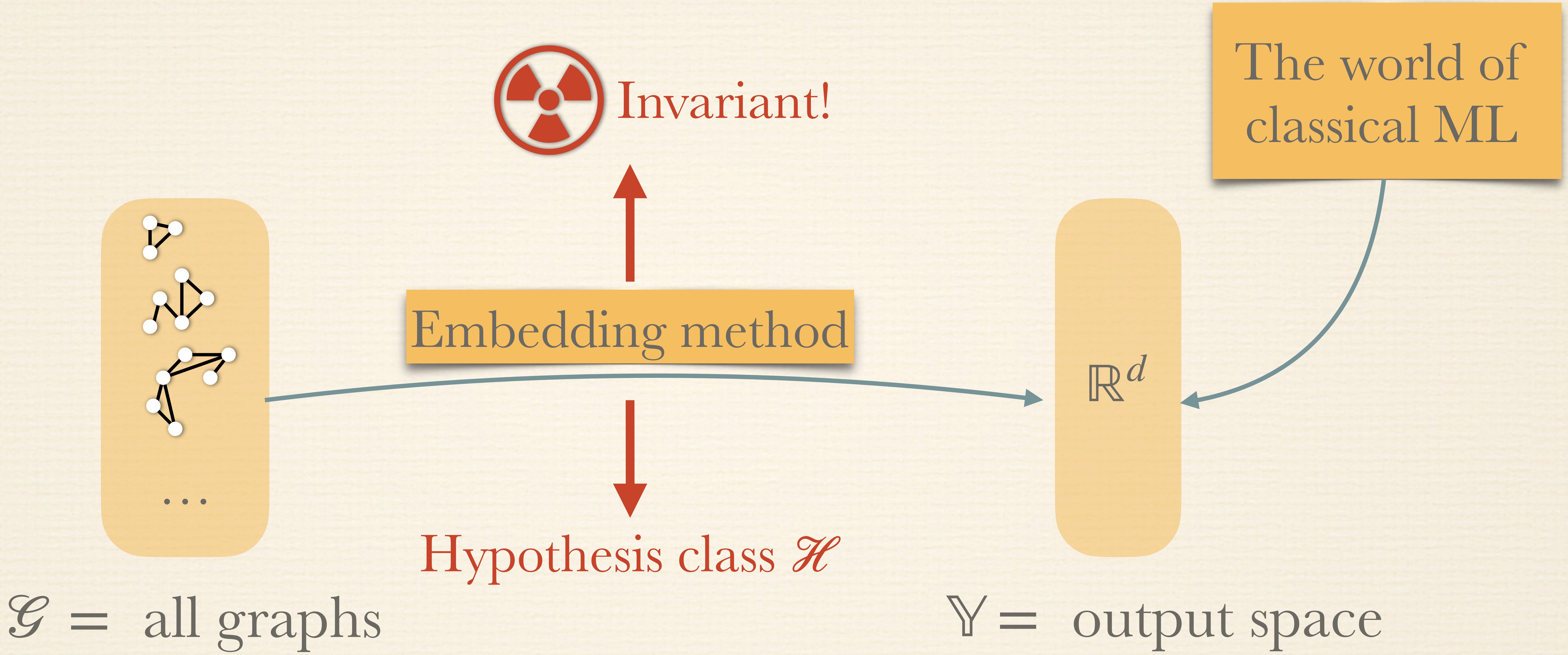
Graph learning: Invariant embeddings



Graph learning: Invariant embeddings



Graph learning: Invariant embeddings

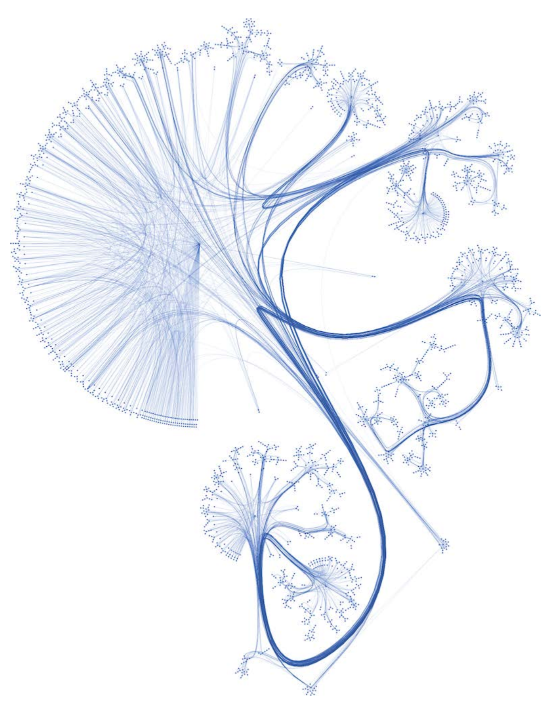


Graph learning: ERM

- Given training set $\mathscr{T}$ and hypothesis class $\mathscr{H}$ of invariant embedding methods $\mathscr{T} := \left\{(G_1, \mathbf{v}_1, y_1), \ldots, (G_\ell, \mathbf{v}_\ell, y_\ell)\right\} \subseteq \mathscr{G} \times \mathscr{V}^p \times \mathbb{Y}$

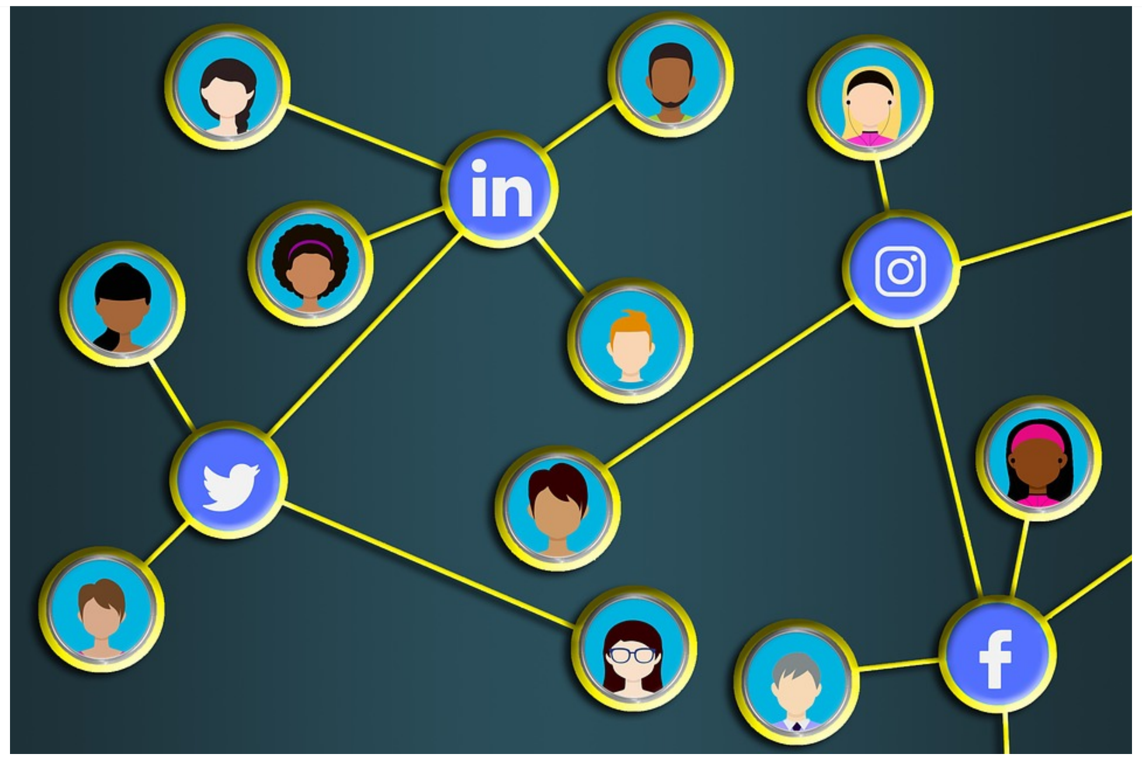
(molecule,yes/no)

Graph classification



(cora, paper, topic)

Vertex classification



(social,p_x,p_y, yes/no)

Link prediction

Graph learning: ERM

- Given training set $\mathcal{T}$ and hypothesis class $\mathcal{H}$ of invariant embedding methods $\mathcal{T} := \{(G_1, \mathbf{v}_1, y_1), \ldots, (G_\ell, \mathbf{v}_\ell, y_\ell)\} \subseteq \mathcal{G} \times \mathcal{V}^p \times \mathbb{Y}$
- Empirical risk minimisation: Find embedding ξ in $\mathcal{H}$ which minimises empirical loss on training set $\mathcal{T}$:

$$\hat{\xi} : \arg\min_{\xi \in \mathcal{H}} \frac{1}{\ell} \sum_{i=1}^{\ell} \mathsf{loss}(\xi(G_i, \mathbf{v}_i), y_i))$$

Hypothesis classes?

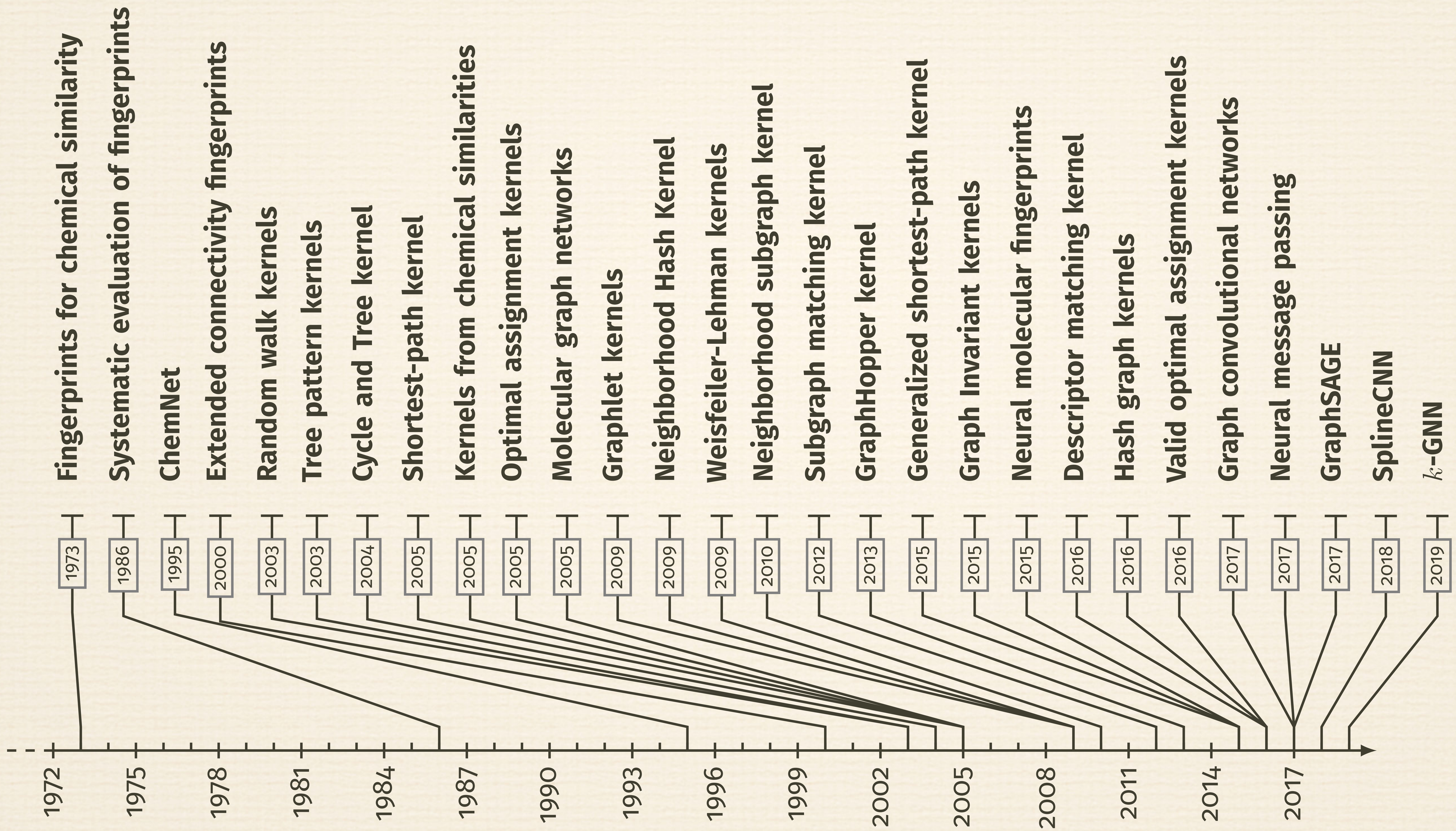


Image: Christopher Morris

Hypothesis classes?

Era of Deep Graph learning

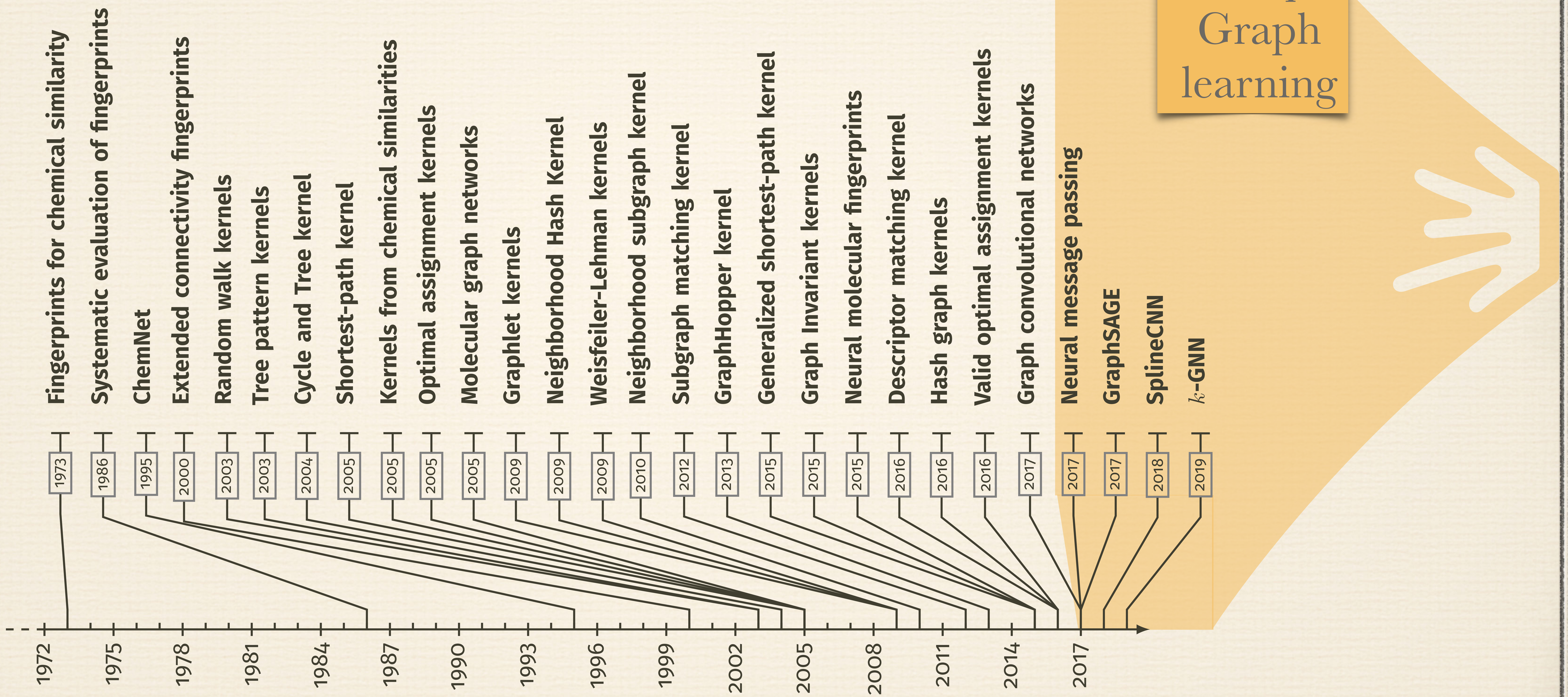


Image: Christopher Morris

“Deep” hypothesis classes

Hypothesis classes: how do they look like?

$$h_i = f_{\text{G-GRU}}\left(x_i, \{h_j : j \to i\}\right) = \mathcal{C}_{\text{G-GRU}}(x_i, \sum_{j \to i} h_j)$$

tion of Eq. (4) does not have an analytical solution, Li et al.
ve scheme:

$$\begin{aligned} h_i^{t+1} &= \mathcal{C}_{\text{G-GRU}}(h_i^t, \bar{h}_i^t), \quad h_i^{t=0} = x_i \ \forall i, \\ \text{where} \ \bar{h}_i^t &= \sum_{j \to i} h_j^t, \end{aligned}$$

) is equal to

$$\begin{aligned} z_i^{t+1} &= \sigma(U_z h_i^t + V_z \bar{h}_i^t) \\ r_i^{t+1} &= \sigma(U_r h_i^t + V_r \bar{h}_i^t) \\ \tilde{h}_i^{t+1} &= \tanh\big(U_h (h_i^t \odot r_i^{t+1}) + V_h \bar{h}_i^t\big) \\ h_i^{t+1} &= (1 - z_i^{t+1}) \odot h_i^t + z_i^{t+1} \odot \tilde{h}_i^{t+1}, \end{aligned}$$

section 4. Consider the matrix $\boldsymbol{X} \in \mathbb{R}^{n \times 2a}$ defined by

$$\boldsymbol{X}_{j,:} = (\mathbf{B}_{j,i_2,:}, \mathbf{B}_{i_1,j,:}), \quad j \in [n]. \tag{7}$$

Our goal is to compute an output tensor $\mathbf{W} \in \mathbb{R}^{n^2 \times b}$, where $\mathbf{W}_{i_1,i_2,:} = u(\boldsymbol{X})$.

Consider the multi-index set $\left\{\boldsymbol{\alpha} \mid \boldsymbol{\alpha} \in [n]^{2a}, |\boldsymbol{\alpha}| \le n\right\}$ of cardinality $b = \binom{n+2a-1}{2a-1}$, and write it in the form $\{(\boldsymbol{\beta}_l, \boldsymbol{\gamma}_l) \mid \boldsymbol{\beta}, \boldsymbol{\gamma} \in [n]^a, |\boldsymbol{\beta}_l| + |\boldsymbol{\gamma}_l| \le n, l \in b\}$.

Now define polynomial maps $\tau_1, \tau_2 : \mathbb{R}^a \to \mathbb{R}^b$ by $\tau_1(x) = (x^{\boldsymbol{\beta}_l} \mid l \in [b])$, and $\tau_2(x) = (x^{\boldsymbol{\gamma}_l} \mid l \in [b])$. We apply τ_1 to the features of $\mathbf{B}$, namely $\mathbf{Y}_{i_1,i_2,l} := \tau_1(\mathbf{B})_{i_1,i_2,l} = (\mathbf{B}_{i_1,i_2,:})^{\boldsymbol{\beta}_l}$; similarly, $\mathbf{Z}_{i_1,i_2,l} := \tau_2(\mathbf{B})_{i_1,i_2,l} = (\mathbf{B}_{i_1,i_2,:})^{\boldsymbol{\gamma}_l}$. Now,

$$\mathbf{W}_{i_1,i_2,l} := (\mathbf{Z}_{:,:,l} \cdot \mathbf{Y}_{:,:,l})_{i_1,i_2} = \sum_{j=1}^n \mathbf{Z}_{i_1,j,l} \mathbf{Y}_{j,i_2,l} = \sum_{j=1}^n \mathbf{B}_{j,i_2,:}^{\boldsymbol{\beta}_l} \mathbf{B}_{i_1,j,:}^{\boldsymbol{\gamma}_l} = \sum_{j=1}^n (\mathbf{B}_{j,i_2,:}, \mathbf{B}_{i_1,j,:})^{(\boldsymbol{\beta}_l, \boldsymbol{\gamma}_l)},$$

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw})$$

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$

the sum, $N(v)$ denotes the neighbors of v i
readout phase computes a feature vector
aph using some readout function R accordi

$$\hat{y} = R(\{h_v^T \mid v \in G\}).$$

$$\psi\Big(M_n H_n^{\text{in}} W_n + U_n H_{n-1}^{\text{in}} W_{n-1} + O_n H_{n+1}^{\text{in}} W_{n+1}\Big), \tag{11}$$

where ψ is an entry-wise activation ($s \mapsto \max\{0, s\}$ for ReLU), $W_n \in \mathbb{R}^{d_n \times m_n}$ are trainable weight matrices and $M_n \in \mathbb{R}^{S_n \times S_n}$, $U_n \in \mathbb{R}^{S_n \times S_{n-1}}$, and $O_n \in \mathbb{R}^{S_n \times S_{n+1}}$ are some choice of adjacency matrices for the simplicial complex. These could be the Hodge Laplacian matrix L_n and the corresponding boundary matrices $B_n^\top$, B_{n+1}, or one of their variants (e.g. normalised).

It is convenient to write the entire layer output in standard form. Using Roth's lemma and concatenating over n we can write (11) as (details in Appendix B)

$$H^{\text{out}} = \psi(W H^{\text{in}}), \tag{12}$$

where $H^{\text{in}} = \text{vec}([H_0^{\text{in}} | H_1^{\text{in}} | \cdots | H_p^{\text{in}}]) \in \mathbb{R}^N$, $N = \sum_{n=0}^p S_n d_n$, $H^{\text{out}} = \text{vec}([H_0^{\text{out}} | H_1^{\text{out}} | \cdots | H_p^{\text{out}}]) \in \mathbb{R}^M$, $M = \sum_{n=0}^p S_n m$, and

$$W = \begin{bmatrix} W_0^\top \otimes M_0 & W_1^\top \otimes O_0 & & & \\ W_0^\top \otimes U_1 & W_1^\top \otimes M_1 & W_2^\top \otimes O_1 & & \\ & W_1^\top \otimes U_2 & W_2^\top \otimes M_2 & W_3^\top \otimes O_2 & \\ & & & & \ddots \end{bmatrix}. \tag{13}$$

is injective. We can make ϵ a learnable parameter or a fixed sc
resentations as

$$h_v^{(k)} = \text{MLP}^{(k)}\left(\left(1 + \epsilon^{(k)}\right) \cdot h_v^{(k-1)} + \sum\nolimits_{u \in \mathcal{N}(v)} h_u^{(k-1)}\right).$$

may exist many other powerful GNNs. GIN is one such exam

$$h_i^{\ell+1} = f_{\text{S-GCN}}^{\ell}\left(\{h_j^\ell : j \to i\}\right) = \text{ReLU}\left(\sum_{j \to i} \eta_{ij} \odot V^\ell h_j^\ell\right)$$

edge gates, and are computed by:

$$\eta_{ij} = \sigma\left(A^\ell h_i^\ell + B^\ell h_j^\ell\right).$$

erative process to solve Eq. (10): At layer ℓ, for $t = 0, 1, ..., T$

$$\begin{aligned} \bar{h}_i^{\ell,t} &= \sum_{j \to i} h_j^{\ell,t}, \\ i_i^{\ell,t+1} &= \sigma(U_i^\ell x_i^\ell + V_i^\ell \bar{h}_i^{\ell,t}) \\ o_i^{\ell,t+1} &= \sigma(U_o^\ell x_i^\ell + V_o^\ell \bar{h}_i^{\ell,t}) \\ \tilde{c}_i^{\ell,t+1} &= \tanh\big(U_c^\ell x_i^\ell + V_c^\ell \bar{h}_i^{\ell,t}\big) \\ f_{ij}^{\ell,t+1} &= \sigma(U_f^\ell x_i^\ell + V_f^\ell h_j^{\ell,t}) \\ c_i^{\ell,t+1} &= i_i^{\ell,t+1} \odot \tilde{c}_i^{\ell,t+1} + \sum_{j \to i} f_{ij}^{\ell,t+1} \odot c_j^{\ell,t+1} \\ h_i^{\ell,t+1} &= o_i^{\ell,t+1} \odot \tanh(c_i^{\ell,t+1}) \end{aligned}$$

and initial conditions: $h_i^{\ell,t=0} = c_i^{\ell,t=0} = 0, \ \forall i, \ell$

How to compare different classes?

- How to compare such embedding classes theoretically?
- How to bring order to the chaos?

How to compare different classes?

- How to compare such embedding classes theoretically?
- How to bring order to the chaos?

1. See graph embedding methods as queries in some query language

2. Analyse expressive power of query language

3. Transfer understanding back to graph learning world

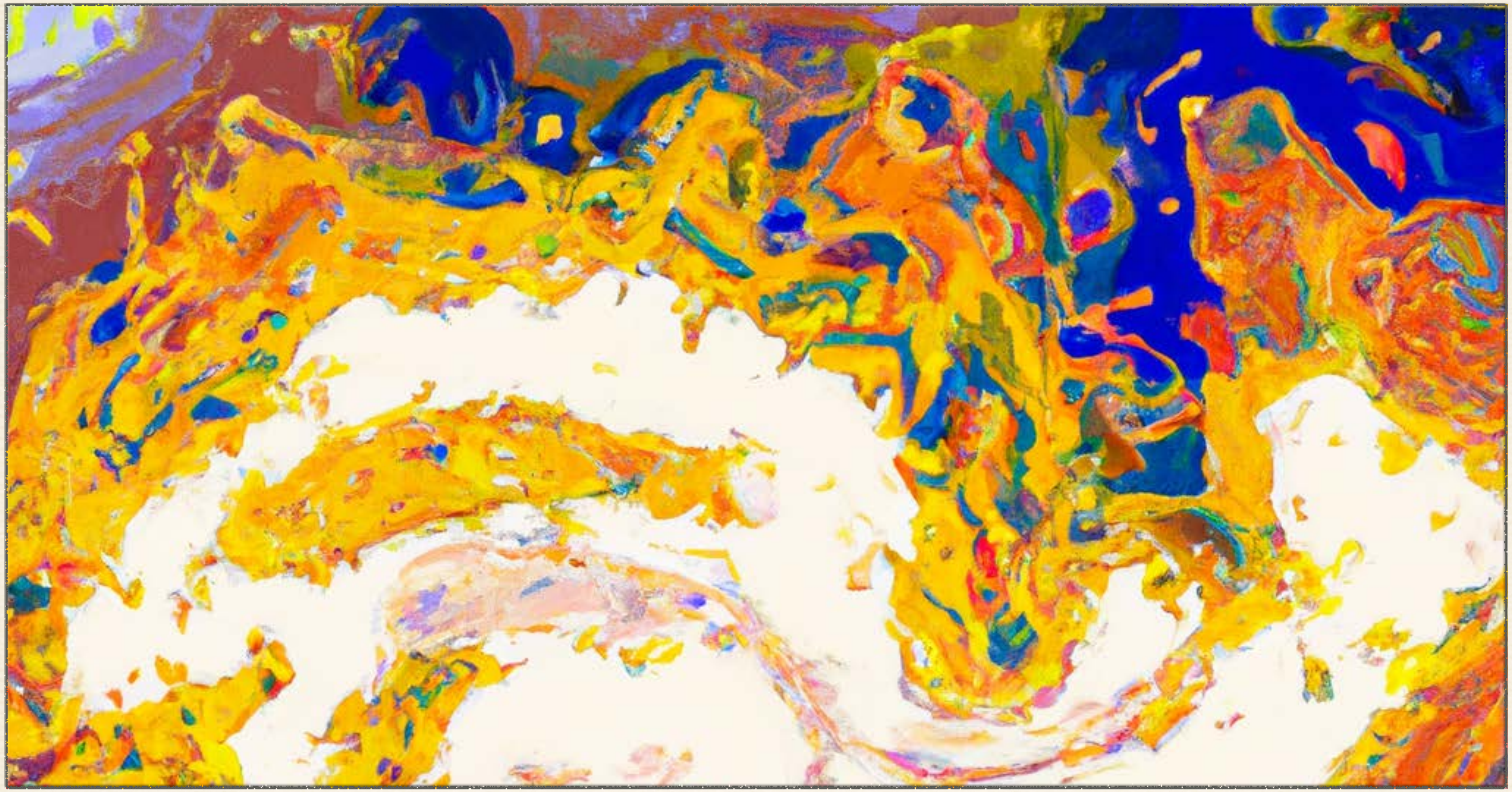
How to compare different classes?

- How to compare such embedding classes theoretically?
- How to bring order to the chaos?

What kind of language?

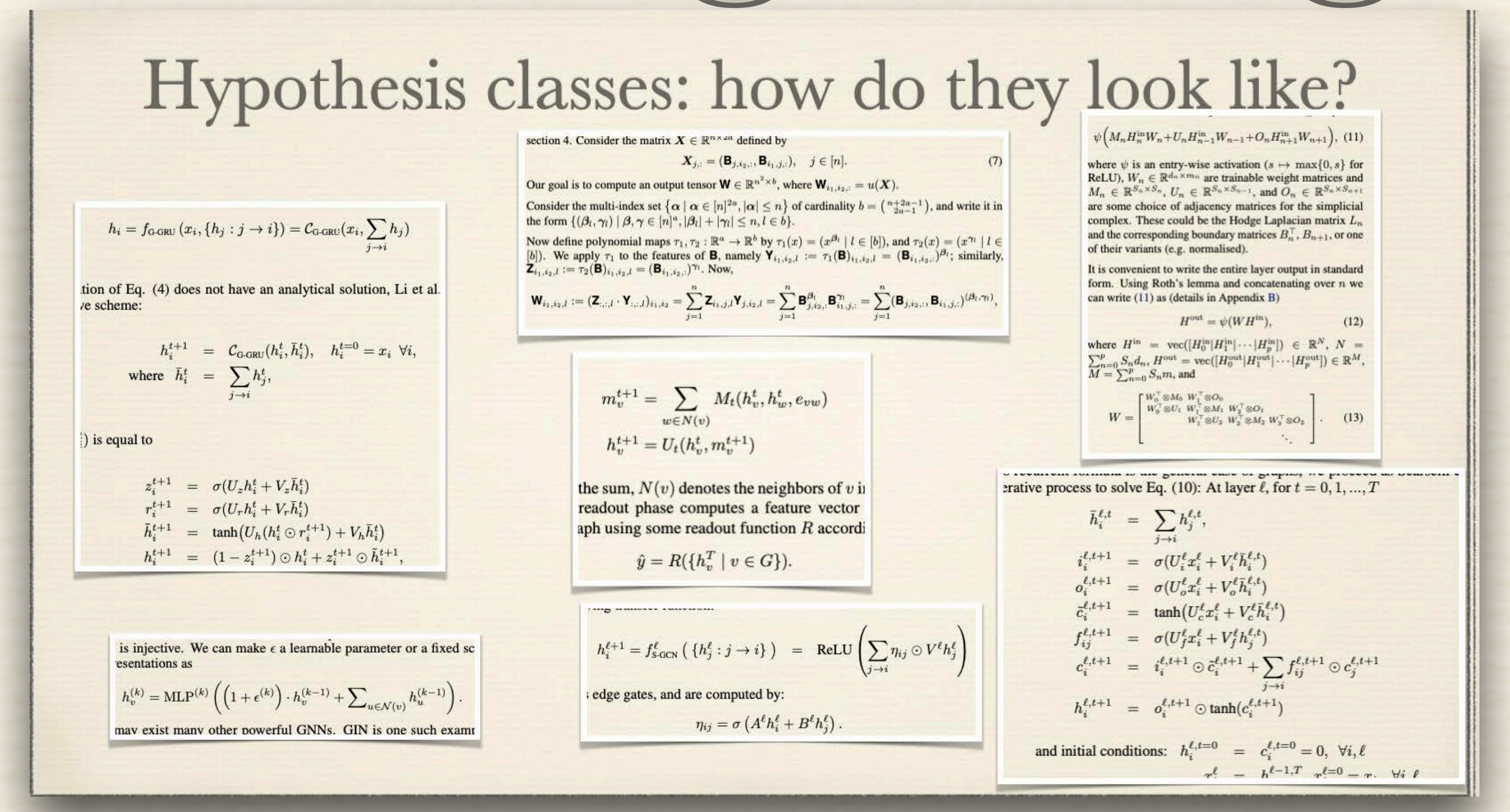
1. See graph embedding methods as queries in some query language
2. Analyse expressive power of query language
3. Transfer understanding back to graph learning world

Expressive power?



Graph Embedding Language

Graph Embedding Language (GEL)



- Most methods are specified in terms of linear algebra computations interleaved with non-linear function applications

- Crucial component is multiplication with adjacency matrix which corresponds to neighbourhood aggregation

Desired language needs function application and aggregation

Graph Neural Networks 101

- Non-linear activation function σ (ReLU, sign, sigmoid, …)
- $\mathbf{F}_G^{(t)} \in \mathbb{R}^{n \times d}$ denotes embedding of vertices in graph G

$$\begin{array}{c} v_1 \\ v_2 \\ v_3 \end{array} \begin{pmatrix} 0.1 & 31 & 8 & 4.03 \\ 5 & 0.03 & 9.7 & -1 \\ -3 & 118 & -63 & 0.204 \end{pmatrix}$$

- Weight matrices $\mathbf{W}_1^{(t)} \in \mathbb{R}^{d \times d}$ and $\mathbf{W}_2^{(t)} \in \mathbb{R}^{d \times d}$ and bias vector $\mathbf{b} \in \mathbb{R}^{1 \times d}$

$\mathbf{F}_G^{(0)} \leftarrow$ Initial hot-one embedding of vertex labels

Matrix form $\mathbf{F}_G^{(t)} := \sigma\left(\mathbf{F}_G^{(t-1)}\mathbf{W}_1^{(t)} + \boxed{\mathbf{A}_G}\mathbf{F}_G^{(t-1)}\mathbf{W}_2^{(t)} + \mathbf{B}^{(t)}\right) \in \mathbb{R}^{n \times d}$

Adjacency matrix

Aggregation over neighbours

GNN 101: Graph embedding

- Weight matrix $\mathbf{W} \in \mathbb{R}^{d \times d}$ and and bias vector $\mathbf{b} \in \mathbb{R}^{1 \times d}$

$$\mathbf{F}_G := \sigma\left(\sum_{v \in V_G} \mathbf{F}_G^{(L)} \mathbf{W} + \mathbf{b}\right) \in \mathbb{R}^{1 \times d}$$

Aggregation over all vertices

- Hypothesis class $\mathscr{H}$ consists of $\xi_\Theta : G \mapsto \mathbf{F}_G$ parametrised by weights and biases $\Theta = \mathbf{W}_1^{(1)}, \ldots, \mathbf{W}_1^{(L)}, \mathbf{W}_2^{(1)} \ldots, \mathbf{W}_2^{((L)}, \mathbf{W}, \mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(L)}, \mathbf{b}$

- Empirical Risk Minimisation: Find best parameters Θ

Graph Embedding Language (GEL)

Syntax

GEL expression

$\varphi(\mathbf{x})$ of dimension d and free variables $\mathbf{x} = \{x_1, \ldots, x_\ell\}$

Semantics

Higher order embedding

$$\xi_\varphi : \mathscr{G} \to (\mathscr{V}^\ell \to \mathbb{R}^d)$$

A simplified version of a query languages with aggregates studied in database theory and it resembles Datalog°

Hella, Libkin, Nurmonen, Wong: *Logics with Aggregates.* (2001)
Abo Khamis, Ngo, Pichler, Suciu, Wang: *Convergence of Datalog over (Pre-) Semiring.* (2022)
G. and Reutter: *Expressiveness and approximation properties of graph neural networks.* (2022)

Atomic GEL expressions

Assigning vertex v to x_i

Atomic expressions

Label: $\varphi(x_i) := \mathsf{Lab}_j(x_i)$ of dim 1 and free var x_i

Edge: $\varphi(x_i, x_j) := E(x_i, x_j)$ of dim 1, free vars x_i, x_j

Equality: $\varphi(x_i, x_j) := \mathbf{1}[x_i = x_j]$ of dim 1, free vars x_i, x_j

Semantics

$\xi_\varphi(G, x_i/v) := j$ th feature of v

$$\xi_\varphi(G, x_i/v, x_j/w) := \begin{cases} 1 & (v, w) \in E_G \\ 0 & \text{otherwise} \end{cases}$$

$$\xi_\varphi(G, x_i/v, x_j/w) := \begin{cases} 1 & v = w \\ 0 & \text{otherwise} \end{cases}$$

GEL: Function Application

Function application: Syntax

Let $\varphi_1(\mathbf{x}_1), \ldots, \varphi_\ell(\mathbf{x}_1)$ be GEL expressions of dim $d_1, \ldots, d_\ell$ and free vars $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$
Let $F : \mathbb{R}^{d_1+\cdots+d_\ell} \to \mathbb{R}^d$ be a function. Then,

$$\varphi(\mathbf{x}) = F(\varphi_1, \ldots, \varphi_\ell)$$

is a GEL expression of dim d and free vars $\mathbf{x} = \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_\ell$

GEL: Function Application

Function application: Syntax

Let $\varphi_1(\mathbf{x}_1), \ldots, \varphi_\ell(\mathbf{x}_1)$ be GEL expressions of dim $d_1, \ldots, d_\ell$ and free vars $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$
Let $F : \mathbb{R}^{d_1+\cdots+d_\ell} \to \mathbb{R}^d$ be a function. Then,

$$\varphi(\mathbf{x}) = F(\varphi_1, \ldots, \varphi_\ell)$$

is a GEL expression of dim d and free vars $\mathbf{x} = \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_\ell$

Semantics

$$\xi_\varphi(G, \mathbf{x}/\mathbf{v}) := F\Big(\underbrace{\xi_{\varphi_1}(G, \mathbf{x}_1/\mathbf{v}_1)}_{\Cap\ \mathbb{R}^{d_1}}, \ldots, \underbrace{\xi_{\varphi_\ell}(G, \mathbf{x}_\ell/\mathbf{v}_\ell)}_{\Cap\ \mathbb{R}^{d_\ell}}\Big) \in \mathbb{R}^d$$

Linear algebra
Activation functions
Anything you want…

GEL: Aggregation

Aggregation: Syntax

Let $\varphi_1(\mathbf{x}, \mathbf{y})$ and $\varphi_2(\mathbf{x}, \mathbf{y})$ be GEL expressions of dim d_1 and d_2 and free vars $\mathbf{x}, \mathbf{y}$. Let Θ be a function mapping bags of vectors in $\mathbb{R}^{d_1}$ to a vector in $\mathbb{R}^d$. Then,

$$\varphi(\mathbf{x}) = \mathsf{agg}_{\mathbf{y}}^{\Theta}[\varphi_1 \mid \varphi_2]$$

is a GEL expression of dim d and free vars $\mathbf{x}$

Semantics

$$\xi_{\varphi}(G, \mathbf{x}/\mathbf{v}) := \Theta\Big(\{\!\{ \underbrace{\xi_{\varphi_1}(G, \mathbf{x}/\mathbf{v}, \mathbf{y}/\mathbf{w})}_{\in\, \mathbb{R}^{d_1}} \mid \quad , \mathbf{w} \in V_G^{|\mathbf{y}|} \}\!\} \Big)$$

GEL: Aggregation

Aggregation: Syntax

Let $\varphi_1(\mathbf{x}, \mathbf{y})$ and $\varphi_2(\mathbf{x}, \mathbf{y})$ be GEL expressions of dim d_1 and d_2 and free vars $\mathbf{x}, \mathbf{y}$. Let Θ be a function mapping bags of vectors in $\mathbb{R}^{d_1}$ to a vector in $\mathbb{R}^{d}$. Then,

$$\varphi(\mathbf{x}) = \mathsf{agg}_{\mathbf{y}}^{\Theta}[\varphi_1 \mid \varphi_2]$$

is a GEL expression of dim d and free vars $\mathbf{x}$

Semantics

$$\xi_\varphi(G, \mathbf{x}/\mathbf{v}) := \Theta\Big(\{\!\!\{ \underbrace{\xi_{\varphi_1}(G, \mathbf{x}/\mathbf{v}, \mathbf{y}/\mathbf{w})}_{\in \mathbb{R}^{d_1}} \mid \underbrace{\xi_{\varphi_2}(G, \mathbf{x}/\mathbf{v}, \mathbf{y}/\mathbf{w}) \neq \mathbf{0}}_{\textbf{guard}}, \mathbf{w} \in V_G^{|\mathbf{y}|} \}\!\!\} \Big)$$

GEL: Aggregation example

$$\varphi = \mathsf{agg}^{\mathsf{sum}}_{x,y,z}\left[\mathbf{1}[y = y] \mid E(x,y) \cdot E(y,z) \cdot E(x,z) \cdot \mathbf{1}[x \neq y] \cdot \mathbf{1}[x \neq z] \cdot \mathbf{1}[y \neq z]\right]$$

$\cdot$ = shorthand for product function application

What does this compute?

GEL: Aggregation example

$$\varphi = \mathsf{agg}^{\mathsf{sum}}_{x,y,z}\left[\mathbf{1}[y = y] \mid E(x,y) \cdot E(y,z) \cdot E(x,z) \cdot \mathbf{1}[x \neq y] \cdot \mathbf{1}[x \neq z] \cdot \mathbf{1}[y \neq z]\right]$$

$\cdot$ = shorthand for product function application

What does this compute? 6 x Triangle count

GEL: Aggregation example

$$\varphi = \mathsf{agg}^{\mathsf{sum}}_{x,y,z}\left[\mathbf{1}[y = y] \mid E(x,y) \cdot E(y,z) \cdot E(x,z) \cdot \mathbf{1}[x \neq y] \cdot \mathbf{1}[x \neq z] \cdot \mathbf{1}[y \neq z]\right]$$

$\cdot$ = shorthand for product function application

What does this compute? 6 x Triangle count

Let us see a more elaborate example

Message Passing Neural Networks

We define $\varphi^{(0)}(x_1) := \mathbf{1}[x_1 = x_1]$
Then for $t > 0$, we get

$$\varphi^{(t)}(x_1) := \mathsf{Upd}^{(t)}\left(\varphi^{(t-1)}(x_1), \mathsf{agg}_{x_2}^{\Theta^{(t)}}\left[\varphi^{(t-1)}(x_2) \mid E(x_1, x_2)\right]\right)$$

For readout layer, we get

$$\varphi := \mathsf{agg}_{x_1}^{\Theta}\left[\varphi^{(L)}(x_1) \mid \mathbf{1}[x_1 = x_1]\right]\Big)$$

dummy guard

edge guarded aggregation

This encompasses the GNNs 101

Gilmer, Schoenholz, Riley, Vinyals, Dahl.: *Neural message passing for quantum chemistry.* (2017)

Fragments of GEL

- $\mathrm{GEL}_k(\Omega, \Theta)$: k variable fragment of GEL with functions in Ω and aggregations in Θ

- $\mathrm{GGEL}_2(\Omega, \Theta)$: 2 variable fragment GEL with edge guarded aggregation only $\varphi(x) = \mathsf{agg}_y^{\Theta}[\varphi_1 \mid E(x, y)]$

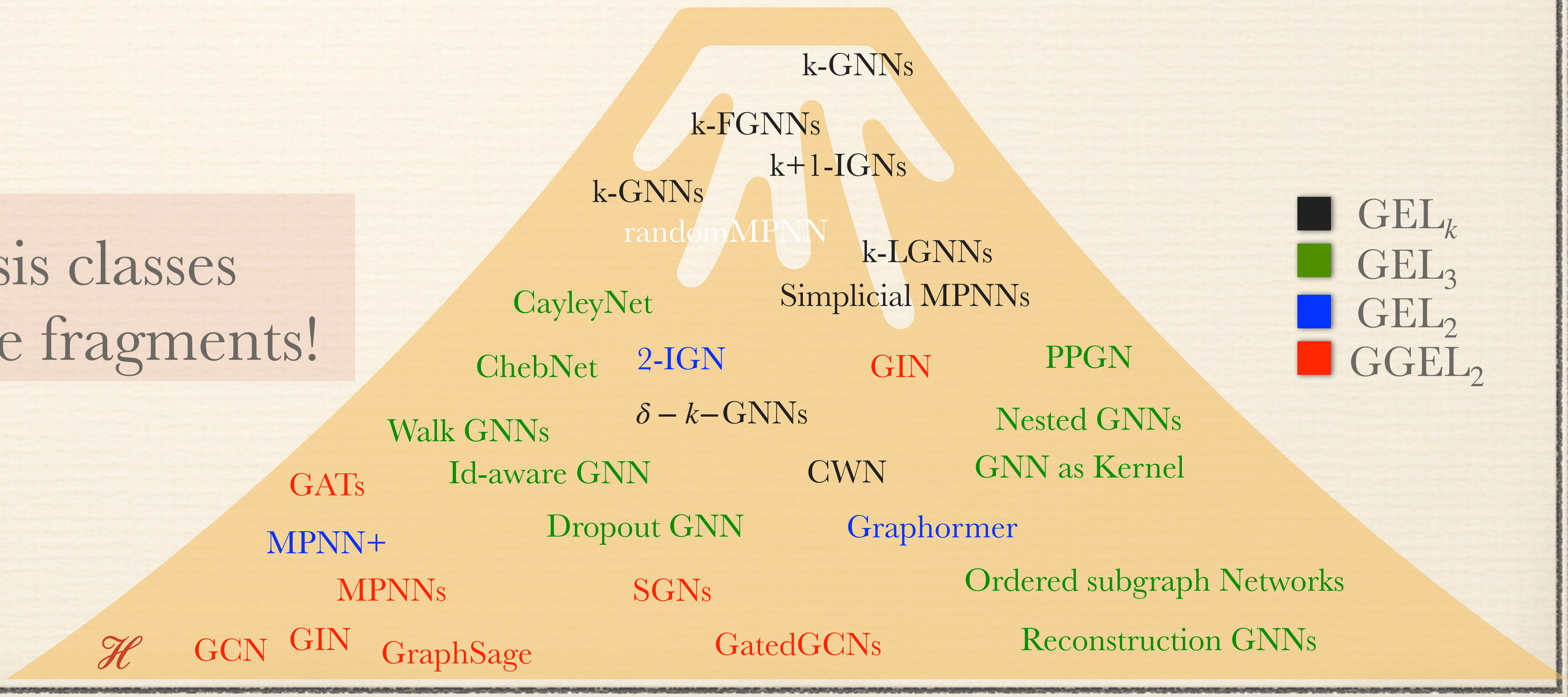
MPNNs without readout phase fit in $\mathrm{GGEL}(\Omega, \Theta)$
MPNNs including readout phase fit in $\mathrm{GEL}_2(\Omega, \Theta)$

Fragments of GEL

- $\mathrm{GEL}_k(\Omega, \Theta)$: k variable fragment of GEL with functions in Ω and aggregations in Θ

- $\mathrm{GGEL}_2(\Omega, \Theta)$: 2 variable fragment GEL with edge guarded aggregation only

Most hypothesis classes fit in one of those fragments!



G., Reutter: *Expressiveness and approximation properties of graph neural networks.* (2022)

Graph Convolutional Networks

Use $D^{-1/2}(I+A)D^{-1/2}$ as propagation matrix

$\varphi(x_1) := F(\mathsf{agg}_{x_2}^{\mathsf{sum}}[\mathbf{1}[x_2 = x_2] \mid E(x_1, x)])$ with $F : \mathbb{R} \to \mathbb{R} : x \mapsto \dfrac{1}{\sqrt{1+x}}$

Hence, $\xi_\phi(G, v) = \dfrac{1}{\sqrt{1 + \deg_G(v)}}$ and we can use

$\psi(x_1, x_2) := \varphi(x_1)(\mathbf{1}[x_1 = x_2] + E(x_1, x_2))\varphi(x_2)$ as the "adjacency" matrix in in the MPNN expressions we have seen before.

GCN$\in$ $\mathrm{GGEL}_2(\Omega, \Theta)$

GCN: Kipf and Welling: *Semi-supervised classification with graph convolutional networks* (2017)

Simplified GCNs

- Uses path information $\mathbf{A}^p\mathbf{F}^{(0)}$ in a single layer.
- For $p = 3$ and for $\varphi^{(0)}(x_1)$ initial feature:
- $\psi(x_1) := \mathsf{agg}_{x_2}^{\mathsf{sum}}\left[\mathsf{agg}_{x_1}^{\mathsf{sum}}\left[\mathsf{agg}_{x_2}^{\mathsf{sum}}\left[\varphi^{(0)}(x_2) \mid E(x_1, x_2)\right] \mid E(x_2, x_1)\right] \mid E(x_1, x_2)\right]$

SGNs $\in$ $\mathrm{GGEL}_2(\Omega, \Theta)$

Wu et al. :*Simplifying Graph Convolutional Networks* (2019)

k-GNNs

$$\xi^{(t)}(x_1, \ldots, x_k) := \sigma\Big(\xi^{(t-1)}(x_1, \ldots, x_k)\mathbf{W}_1^{(t)} + \big(\sum_{i=1}^{k} \mathsf{agg}_y^{\mathsf{sum}}[\xi^{(t)}(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_k)]\big)\mathbf{W}_2^{(t)} \Big)$$

k-vertex embedding

Activation Function

Weight matrix

Global aggregation

Weight matrix

k-GNNs $\in \mathrm{GEL}_{k+1}(\Omega, \Theta)$

Morris et al.: *Weisfeiler and Leman go neural: Higher-order graph neural networks.* (2019)

k-Folklore GNNs (k-FGNs)

$$\xi^{(t)}(x_1, \ldots, x_k) := \mathsf{MLP}_1^{(t)}\Big(\mathsf{agg}_y^{\mathsf{sum}}\big[\prod_{i=1}^{k} \mathsf{MLP}_2^{(t)}(\xi^{(t-1)}(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_k)\big]\Big)$$

k-vertex embedding

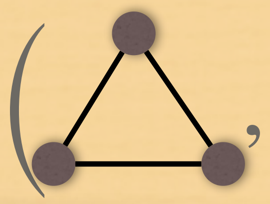
Global aggregation

k-FGNs $\in \mathrm{GEL}_k(\Omega, \Theta)$

Maron et al.: *Provably powerful graph networks* (2019)

W. Azizian and M. Lelarge. *Characterizing the expressive power of invariant and equivariant graph neural networks* (2021)

Subgraph count GNNs

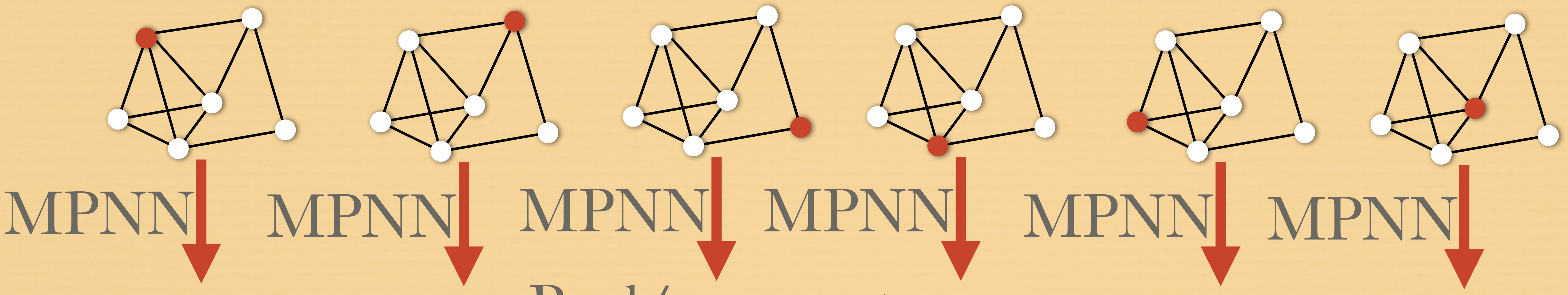
- Use count of subgraphs to augment MPNNs
  - homomorphism count $\mathsf{hom}(P^r, G^v)$ for rooted motif P,
  - subgraph iso count $\mathsf{sub}(P^r, G^v)$ for rooted motif P
- If motif has tree width k then $\mathsf{hom}(P^r, G^v)$ can be computed using k+1 variables.
- For example, $(G, v) \mapsto \mathsf{hom}\Big($  $, G^v\Big)$ can be expressed as

$$\varphi(x_1) := \mathsf{agg}^{\mathsf{sum}}_{x_2}\mathsf{agg}^{\mathsf{sum}}_{x_3}[E(x_1, x_2)E(x_1, x_3)E(x_2, x_3)(\mathbf{1}[x_1 = x_1] - \mathbf{1}[x_1 = x_2])$$
$$(\mathbf{1}[x_1 = x_1] - \mathbf{1}[x_1 = x_3])(\mathbf{1}[x_1 = x_1] - \mathbf{1}[x_2 = x_3])]$$

Tree width k $\mapsto \mathsf{GEL}_{k+1}(\Omega, \Theta)$

Bouritsas et al.: *Improving graph neural network expressivity via subgraph isomorphism counting* (2020)
Barceló et al.: *Graph neural networks with local graph parameters.* (2021)

Subgraph GNNs: vertices



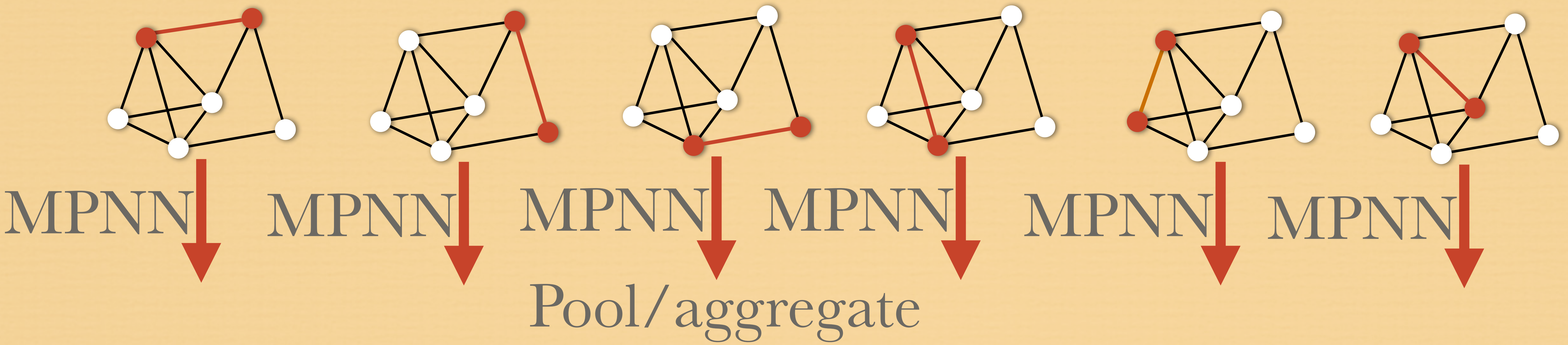
Pool/aggregate

$$\varphi^{(0)}(x_1, x_2) := \mathbf{1}[x_1 = x_2]$$
$$\varphi^{(t)}(x_1, x_2) := \mathsf{Upd}^{(t)}\Big(\varphi^{(t-1)}(x_1, x_2), \mathsf{agg}^{\Theta}_{x_3}[\varphi^{(t-1)}(x_1, x_3) \,|\, E(x_2, x_3)]\Big)$$

3 variables $\mapsto$ $\mathrm{GEL}_3(\Omega, \Theta)$

Bevilacqua et al: *Equivariant subgraph aggregation network* (2022)
Cotta et al.: *Reconstruction for powerful graph representations* (2021)
Bevilacqua et al.: *Understanding and extending subgraph GNNs by rethinking their symmetries* (2022)
Huang et al.: *Boosting the cycle counting power of graph neural networks with I2-GNNs* (2022)
Papp et al.: *DropGNN: Random dropouts increase the expressiveness of graph neural networks.* (2021)
Qian et al.: *Ordered subgraph aggregation networks.* (2022)
You et al.: *Identity-aware graph neural networks.* (2021)
Zhang and P. Li. *Nested graph neural networks* (2021)
Zhao et al.: *From stars to subgraphs: Uplifting any GNN with local structure awareness* (2022)

Subgraph GNNs: edges



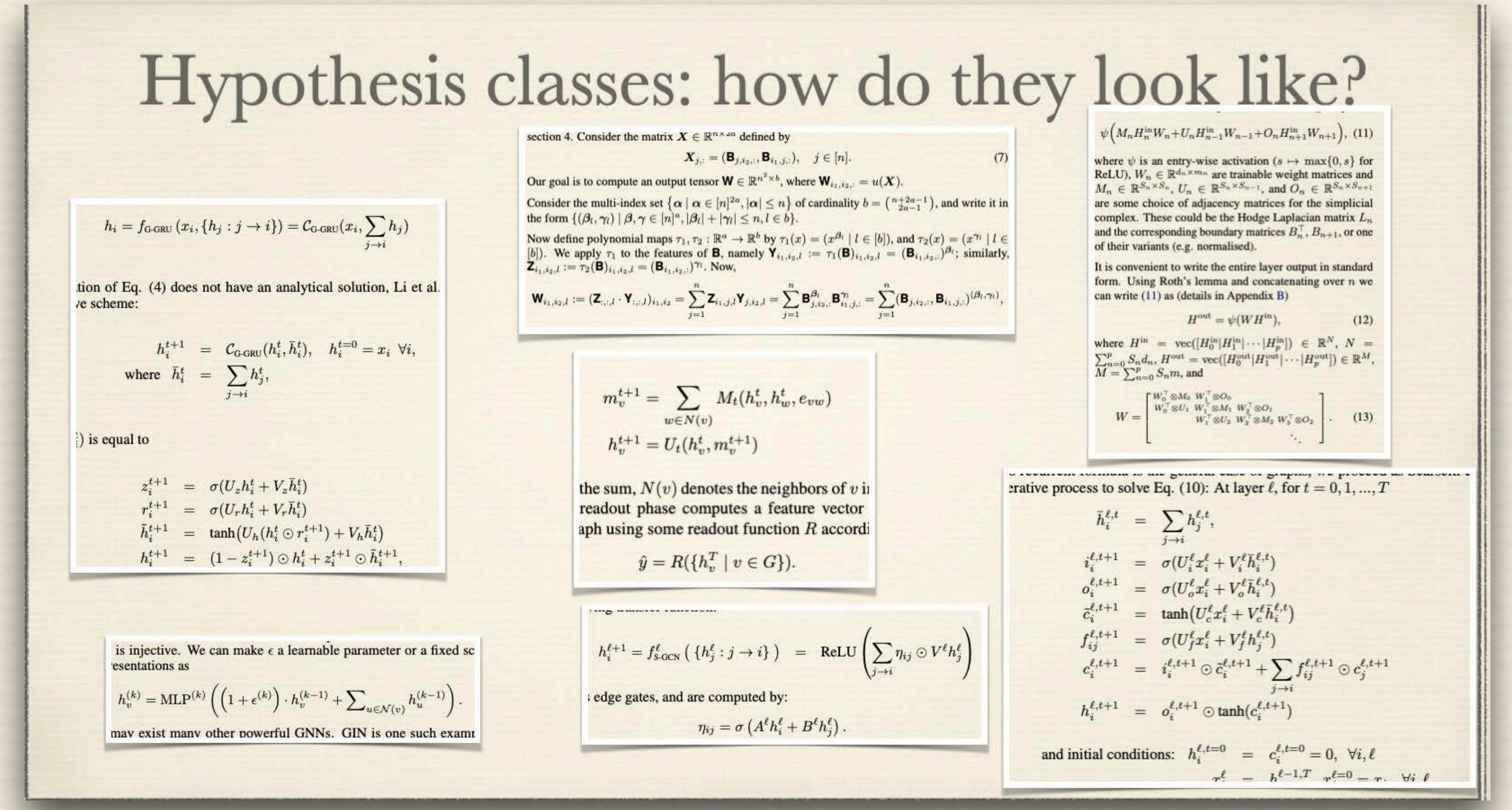
$$\varphi^{(t)}(x_1, x_2, x_3) := \mathsf{Upd}^{(t)}\Big(\varphi^{(t-1)}(x_1, x_2, x_3), \mathsf{agg}^{\Theta}_{x_4}[\varphi^{(t-1)}(x_1, x_2, x_4) \mid E(x_3, x_4)]\Big)$$

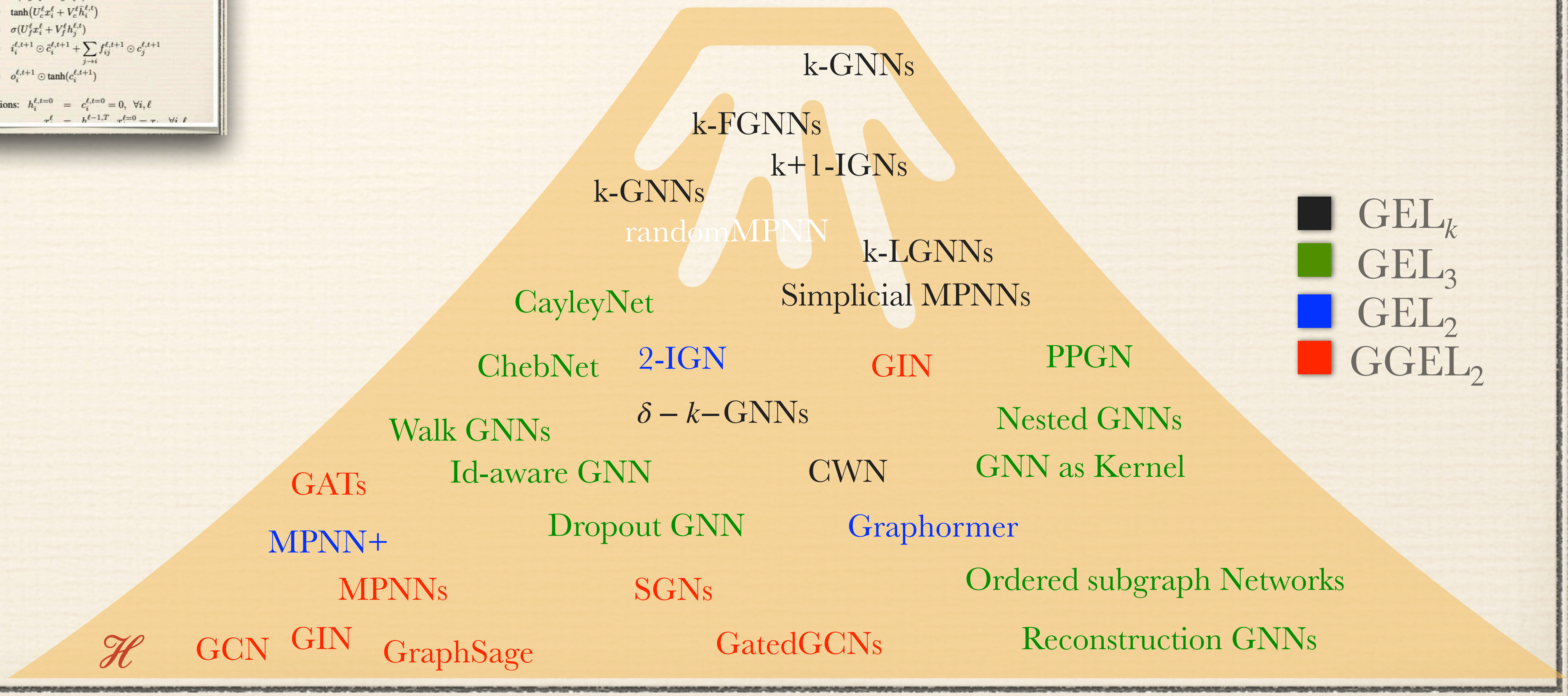
4 variables $\mapsto \mathrm{GEL}_4(\Omega, \Theta)$

Bevilacqua et al: *Equivariant subgraph aggregation network* (2022)
Cotta et al.: *Reconstruction for powerful graph representations* (2021)
Bevilacqua et al.: *Understanding and extending subgraph GNNs by rethinking their symmetries* (2022)
Huang et al.: *Boosting the cycle counting power of graph neural networks with I2-GNNs* (2022)
Papp et al.: *DropGNN: Random dropouts increase the expressiveness of graph neural networks.* (2021)
Qian et al.: *Ordered subgraph aggregation networks.* (2022)
You et al.: *Identity-aware graph neural networks.* (2021)
Zhang and P. Li. *Nested graph neural networks* (2021)
Zhao et al.: *From stars to subgraphs: Uplifting any GNN with local structure awareness* (2022)

Takeaway message #1:

Classification in terms of number of variables





G., Reutter: *Expressiveness and approximation properties of graph neural networks.* (2022)

How to compare different classes?

- How to compare such embedding classes theoretically?
- How to bring order to the chaos?

Which language?

1. See graph embedding methods as queries in some query language

2. Analyse expressive power of query language

3. Transfer understanding back to graph learning world

How to compare different classes?

- How to compare such embedding classes theoretically?
- How to bring order to the chaos?

Which language?

1. See graph embedding methods as queries in some query language

GEL_k
GGEL_2

2. Analyse expressive power of query language

3. Transfer understanding back to graph learning world

Expressive power

Distinguishing power

- Which inputs can be separated/distinguished by embeddings in $\mathscr{H}$?
- Captured by the following equivalence relation on $\mathscr{G} \times \mathscr{V}^p$:

$$\rho(\mathscr{H}) := \{(G, \mathbf{v}, H, \mathbf{w}) \mid \forall \xi \in \mathscr{H} : \xi(G, \mathbf{v}) = \xi(H, \mathbf{w})\}$$

Distinguishing power

- Which inputs can be separated/distinguished by embeddings in $\mathscr{H}$?
- Captured by the following equivalence relation on $\mathscr{G} \times \mathscr{V}^p$:

$$\rho(\mathscr{H}) := \{(G, \mathbf{v}, H, \mathbf{w}) \mid \forall \xi \in \mathscr{H} : \xi(G, \mathbf{v}) = \xi(H, \mathbf{w})\}$$

- Strongest power: $\mathscr{H}$ powerful enough to detect non-isomorphic graphs: $\rho(\mathscr{H})$ only contains isomorphic pairs
- Weakest power: $\mathscr{H}$ cannot differentiate any two graphs: $\rho(\mathscr{H})$ contains all pairs of graphs.

Distinguishing power

- Allows for comparing different classes of embeddings methods

$$\rho(\text{methods}_1) \subseteq \rho(\text{methods}_2)$$

methods$_1$ is more powerful than methods$_2$
methods$_2$ is bounded by methods$_1$ in power

$$\rho(\text{methods}_1) = \rho(\text{methods}_2)$$

Both methods are as powerful

- Allows for comparing embedding methods with algorithms, logic, … on graphs

Expressive power in ML community

- Focus has been on characterising the distinguishing power of classes $\mathscr{H}$ of embedding methods.
- Hopefully, characterisations of $\rho(\mathscr{H})$ shed light on what graph properties a learning method in $\mathscr{H}$ can detect/use.

Logic

- First-order logic with k variables and counting quantifiers (C_k).

k=2

$$\varphi(x) = \exists^{\leq 5} y \left(E(x,y) \land \exists^{\geq 2} x \left(E(y,x) \land L_a(x) \right) \right)$$

binary edge predicate   unary label predicate

- Given graph G, vertex $v \in V_G$ satisfies φ: It has at most 5 neighbours
  each with at least two neighbours labeled "a"

- Guarded fragment GC_2 of C_2

  only existential quantification for the form $\exists^{\geq n} y (E(x,y) \land \varphi(y))$

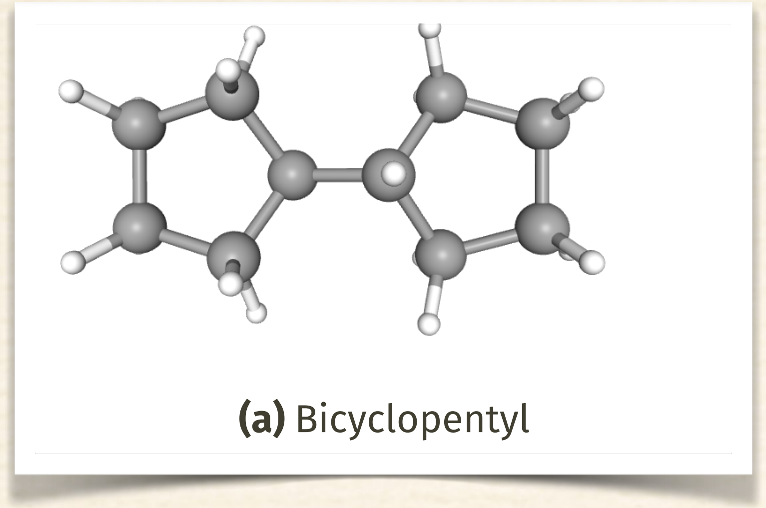
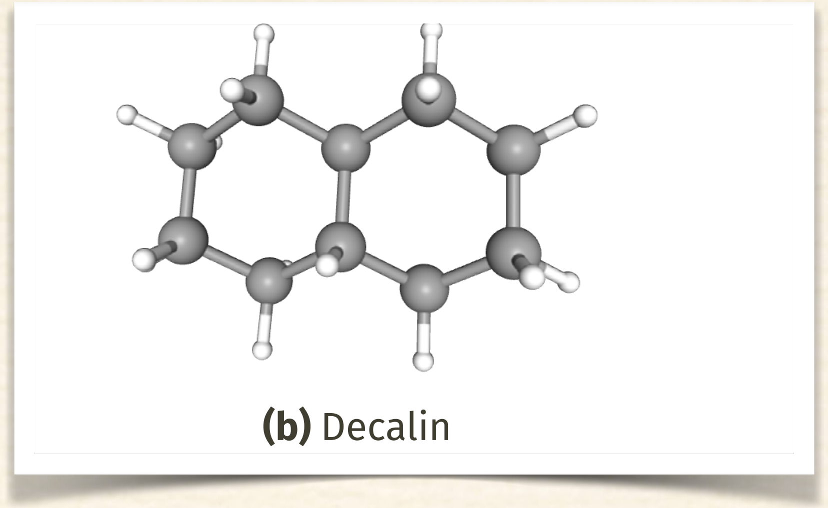
Logic

- We can consider $\rho(\mathsf{C}_k)$ and $\rho(\mathsf{GC}_2)$

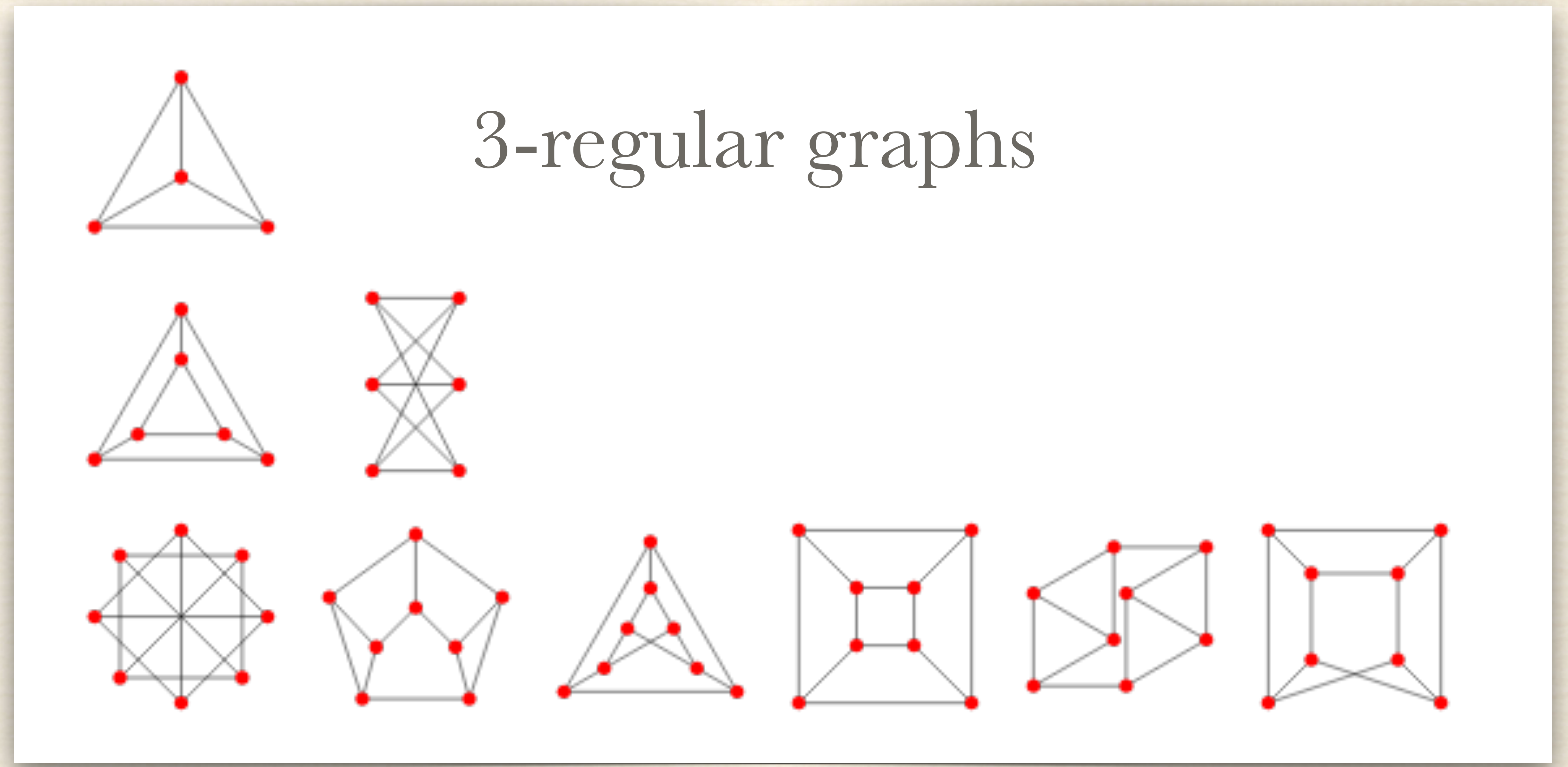
$$\rho(\mathsf{C}_k) := \{(G, \mathbf{v}, H, \mathbf{w}) \mid \forall \varphi \in \mathsf{C}_k : (G, \mathbf{v}) \models \varphi \iff (H, \mathbf{w}) \models \varphi)\}$$

- The distinguishing power of these logics is well understood

$\rho(C_2)$

- Cannot distinguish d-regular graphs
- Cannot count cycles (triangles)
- Only tree information



Arvind et al.: *On the power of color refinement* (2015)
Images: Wolfram MathWorld, Christopher Morris

Expressive power of GEL: Main result

- The following results follow from standard analysis of aggregate query languages: all real number arithmetic can be eliminated.

Theorem (Xu et al. 2019, Morris et al. 2019, G. and Reutter 2022)

$$\rho(\mathrm{GGEL}(\Omega, \Theta)) = \rho(\mathrm{GC}_2)$$

Theorem (G. and Reutter 2022)

$$\rho(\mathrm{GEL}_k(\Omega, \Theta)) = \rho(\mathsf{C}_k)$$

- Lower bounds: Ω contains linear combinations, concatenation, product (or activation functions) and Θ contains summation

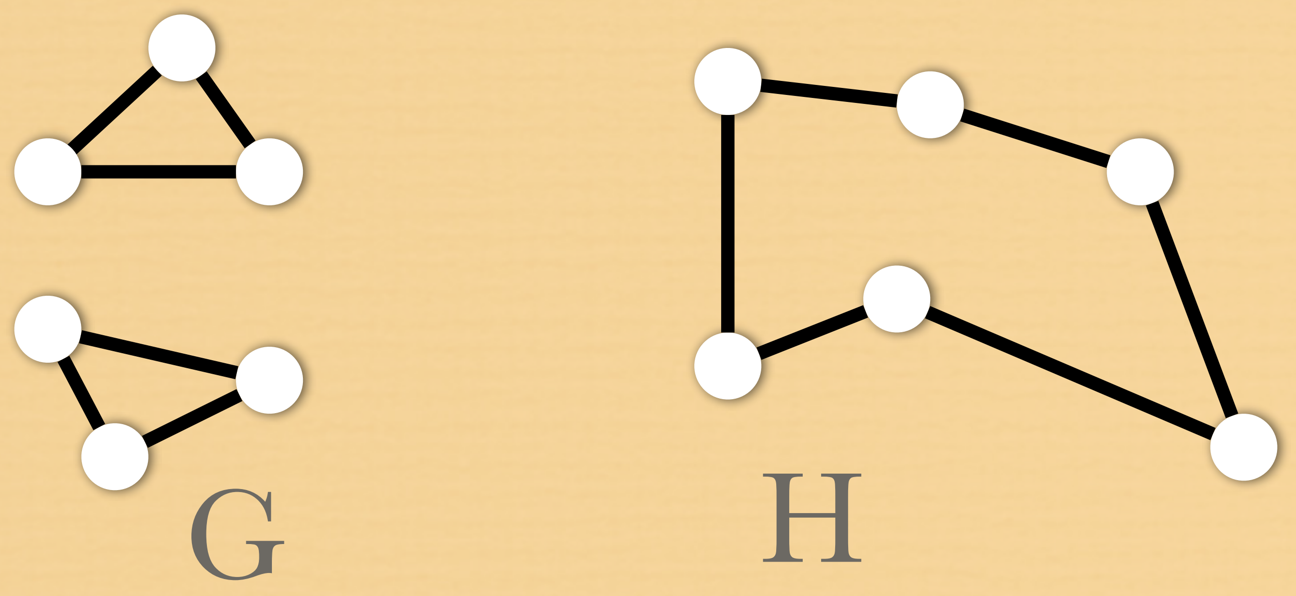
Xu, Hu, Leskovec, Jegelka: *How powerful are graph neural networks?* (2019)
Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, Grohe: *Weisfeiler and Leman go neural: Higher-order graph neural networks.* (2019
Hella, Libkin, Nurmonen, Wong: *Logics with Aggregates.* (2001)
Cai, Fürer, Immerman: *An optimal lower bound on the number of variables for graph identification.* (1992)
G., Reutter: *Expressiveness and approximation properties of graph neural networks.* (2022)
M. Grohe: *The logic of graph neural networks.* (2021)

GNN 101

Theorem (Morris et al. 2019)

$$\rho(\mathrm{GNN101}) = \rho(\mathrm{C}_2)$$

GNN 101s, MPNNs are pretty weak



Can we train a GNN 101 which embeds G differently from H?

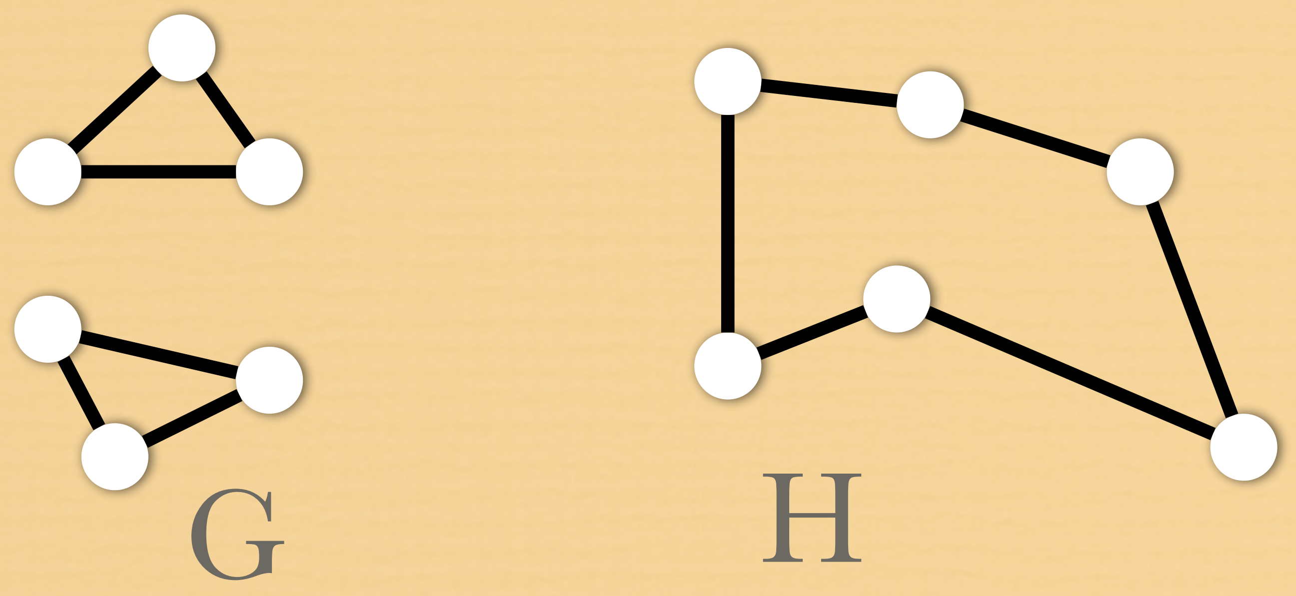
Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, Grohe: *Weisfeiler and Leman go neural: Higher-order graph neural networks.* (2019

GNN 101

Theorem (Morris et al. 2019)

$$\rho(\mathrm{GNN101}) = \rho(\mathrm{C}_2)$$

GNN 101s, MPNNs are pretty weak



Can we train a GNN 101 which embeds G differently from H?

NO!

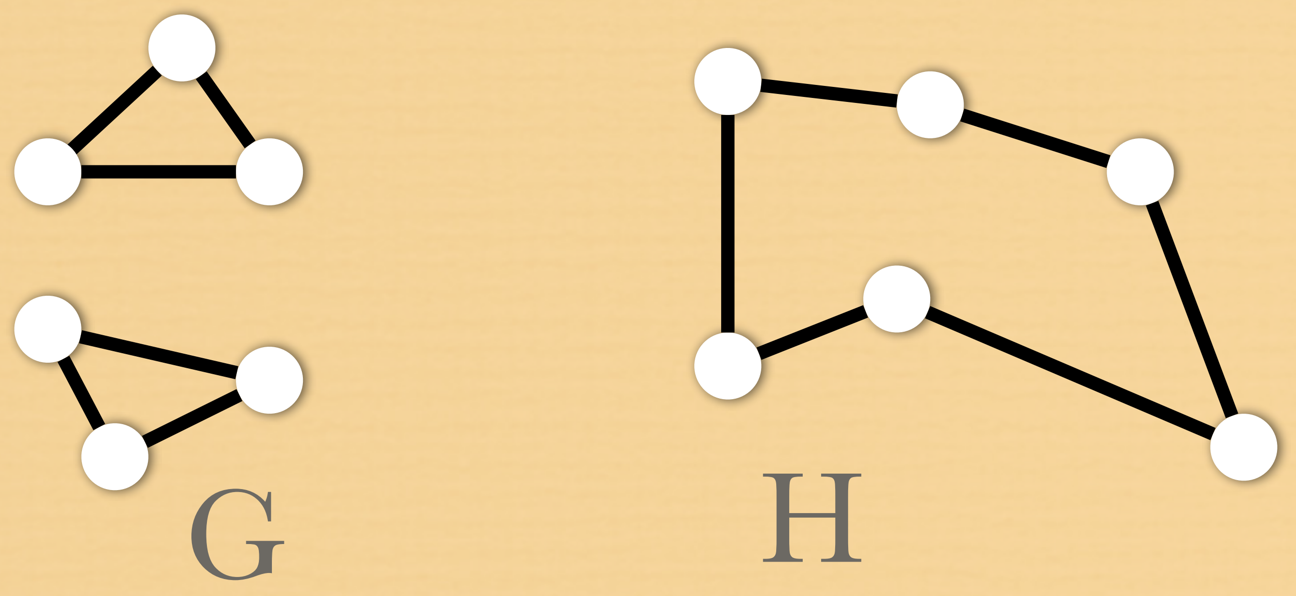
Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, Grohe: *Weisfeiler and Leman go neural: Higher-order graph neural networks.* (2019

GNN 101

Theorem (Morris et al. 2019)

$$\rho(\mathrm{GNN101}) = \rho(\mathrm{C}_2)$$

GNN 101s, MPNNs are pretty weak



Can we train a GNN 101 which embeds G differently from H?

NO!

G and H are known to be indistinguishable by C_2

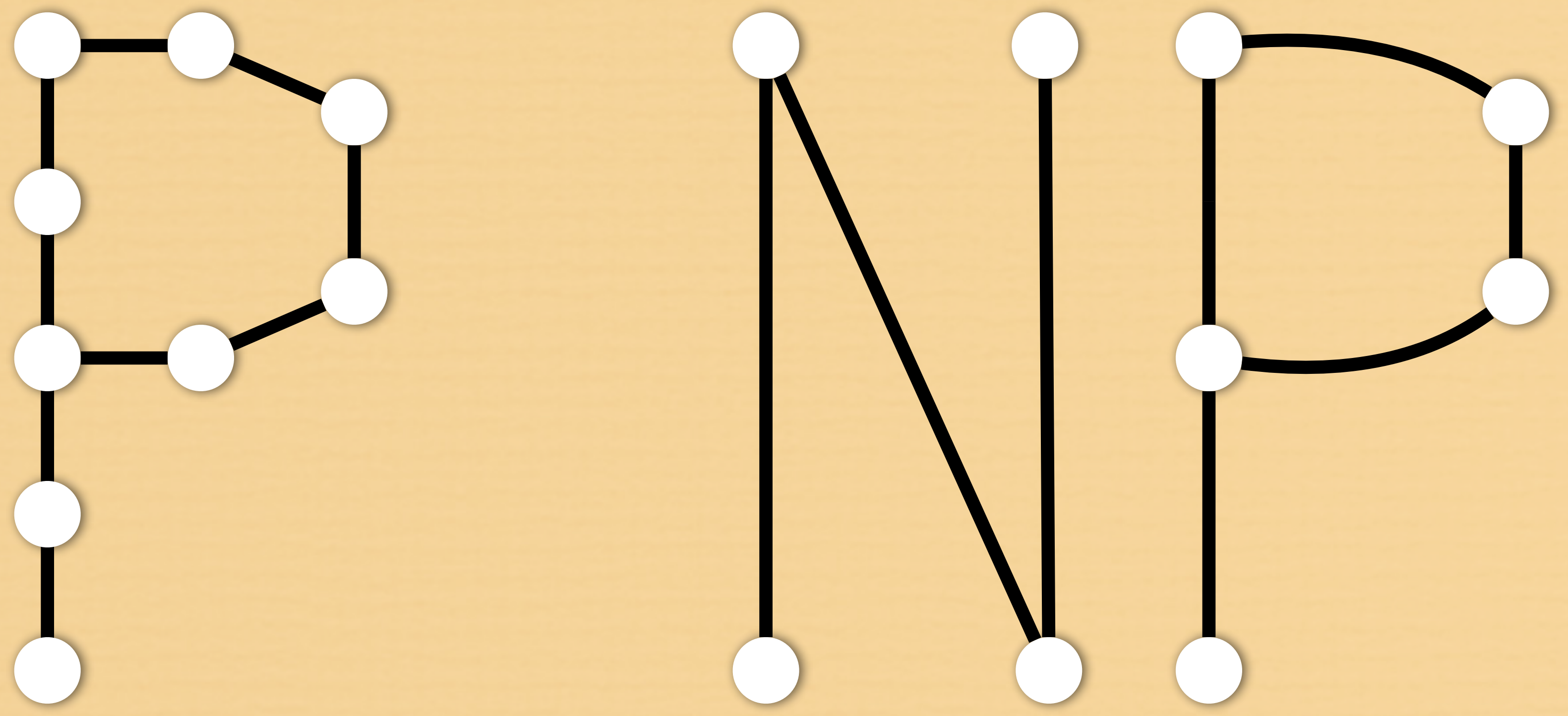
$$\Rightarrow (G, H) \in \rho(\mathsf{C}_2) = \rho(\mathrm{GNN101})$$

Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, Grohe: *Weisfeiler and Leman go neural: Higher-order graph neural networks.* (2019

GNN 101

Theorem (Morris et al. 2019)

$$\rho(\text{GNN101}) = \rho(\text{C}_2)$$



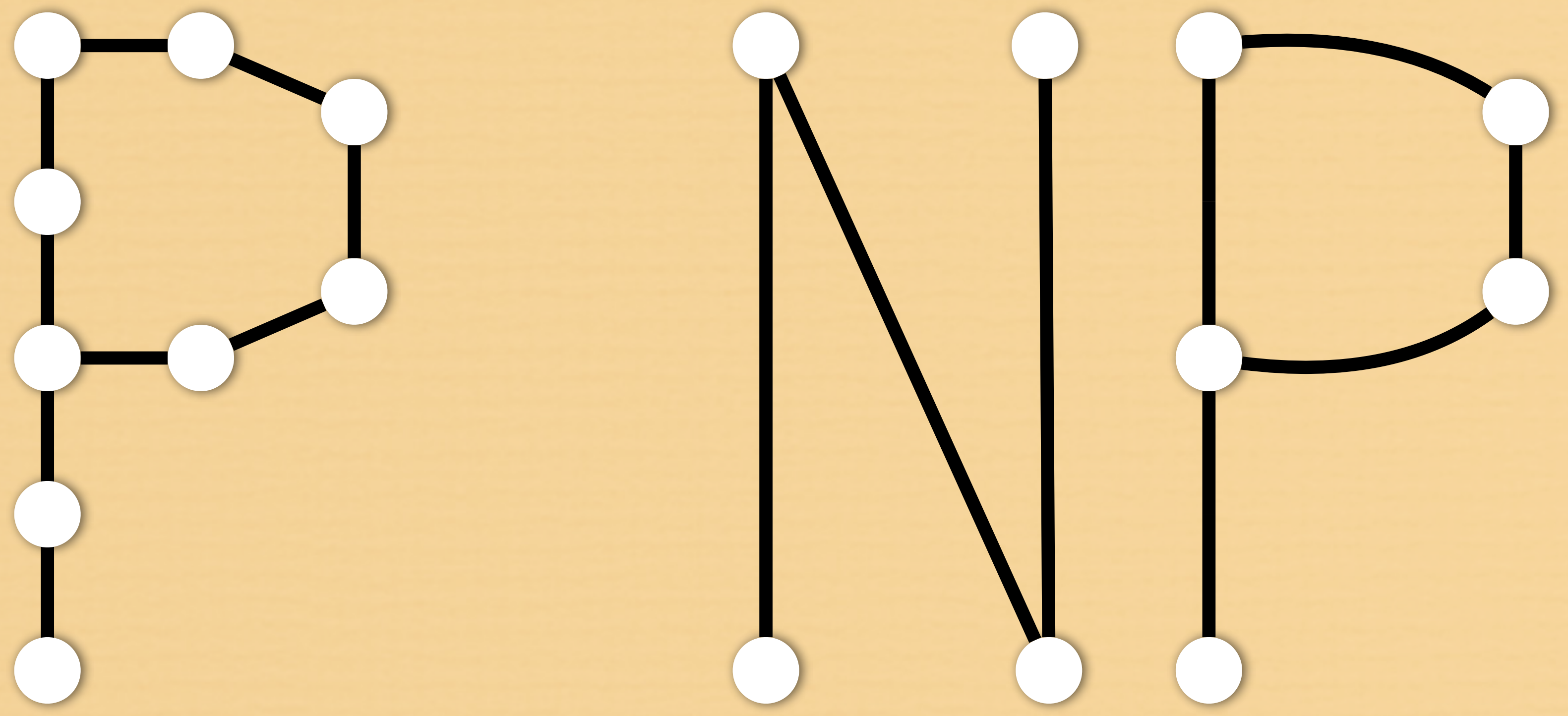
Can we train a GNN101 such that P embeds differently from NP?

Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, Grohe: *Weisfeiler and Leman go neural: Higher-order graph neural networks.* (2019

GNN 101

Theorem (Morris et al. 2019)

$$\rho(\text{GNN101}) = \rho(\text{C}_2)$$



Can we train a GNN101 such that P embeds differently from NP?

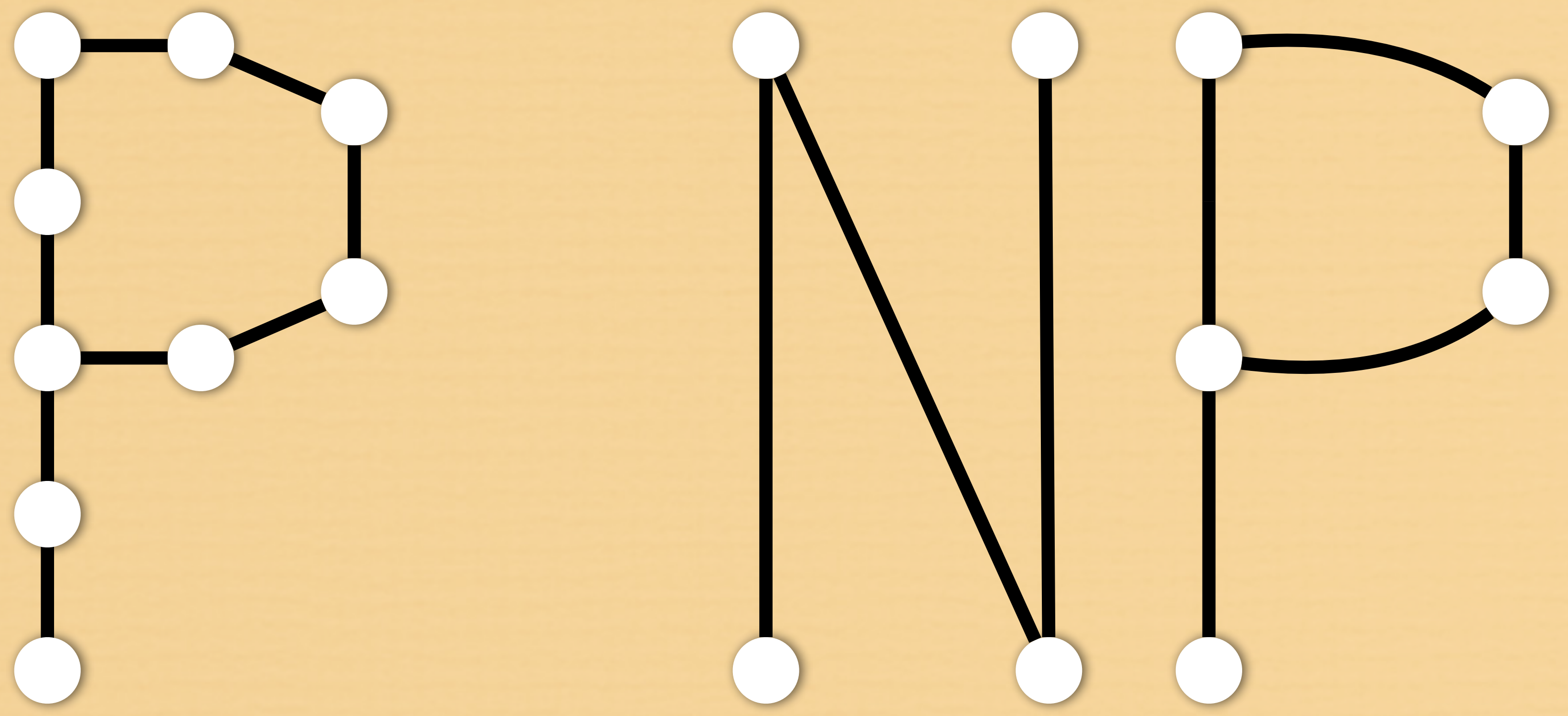
YES!

Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, Grohe: *Weisfeiler and Leman go neural: Higher-order graph neural networks.* (2019

GNN 101

Theorem (Morris et al. 2019)

$$\rho(\mathrm{GNN101}) = \rho(\mathsf{C}_2)$$



Can we train a GNN101 such that P embeds differently from NP?

YES!

single degree one node

P satisfies $\exists^{=1} x \exists^{=1} y\, E(x, y)$ but NP does not

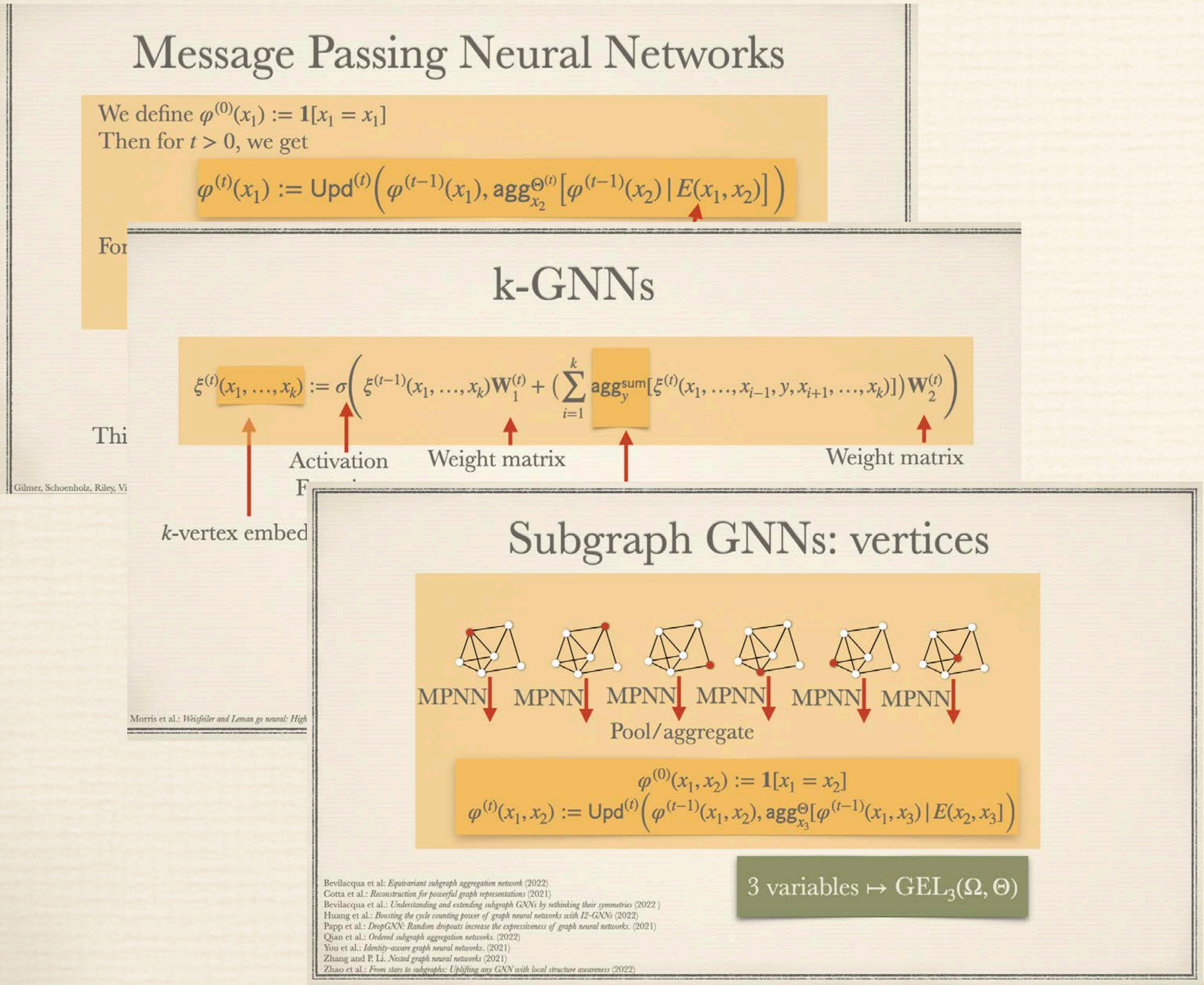
$$\Downarrow$$

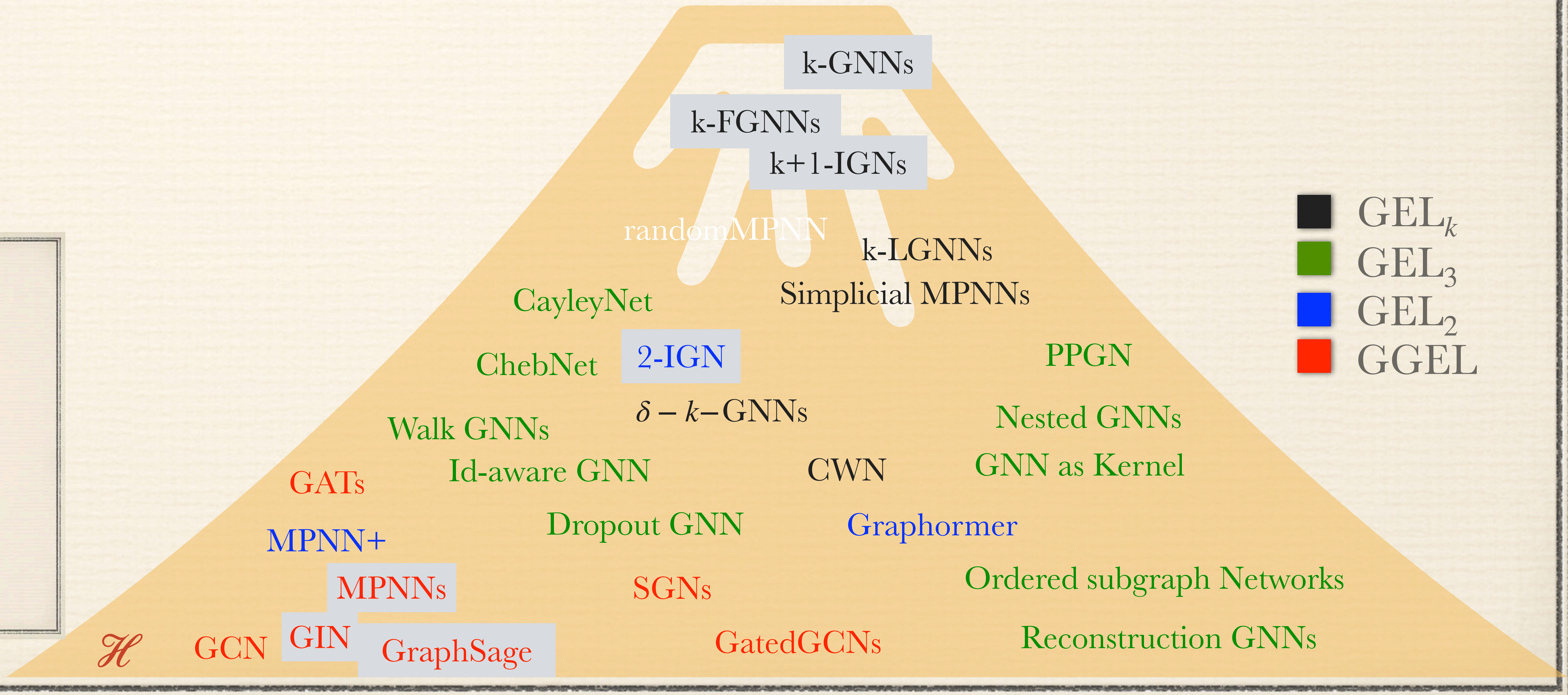
$$(P, NP) \notin \rho(\mathsf{C}_2) \Rightarrow (P, NP) \notin \rho(\mathrm{GNN101})$$

Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, Grohe: *Weisfeiler and Leman go neural: Higher-order graph neural networks.* (2019

Consequences

- If embedding method M can be cast in $\mathrm{GEL}_k(\Omega, \Theta)$ then $\rho(\mathsf{C}_k) \subseteq \rho(M)$
- If embedding method M can also encode formulas in C_k then $\rho(\mathsf{C}_k) \supseteq \rho(M)$





What else can we say?

$\rho(\text{MPNNs}) = \rho(\mathsf{C}_2)$

What else can we say?

$\rho(\text{MPNNs}) = \rho(\mathsf{C}_2)$

Other - more insightful - characterisations?

What else can we say?

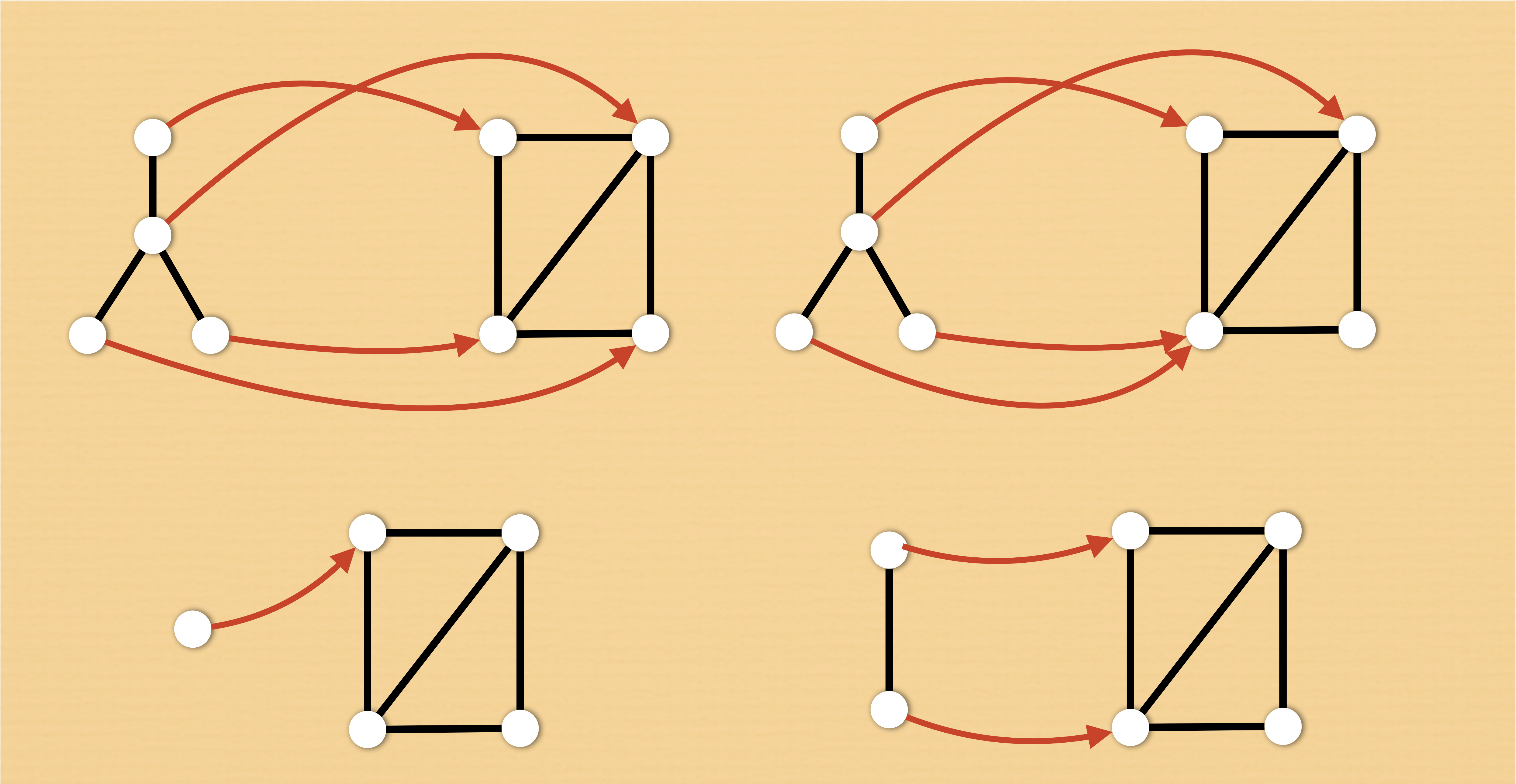
$$\rho(\text{MPNNs}) = \rho(\mathsf{C}_2)$$

Other - more insightful - characterisations?

homomorphism counts

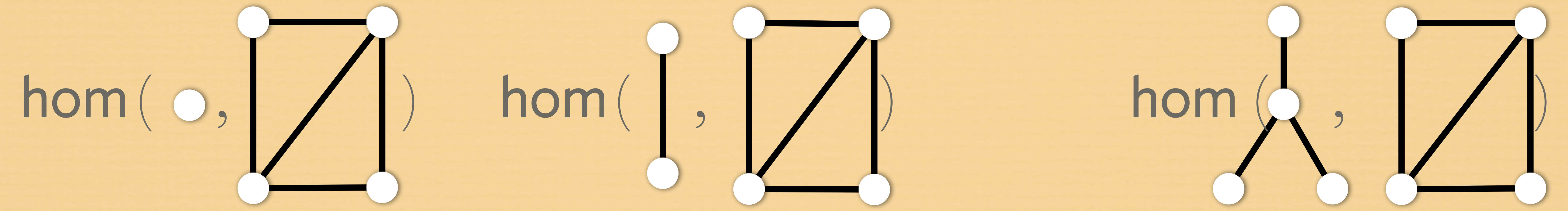
Homomorphisms

- Let $P = (V_P, E_P, L_P)$ and $G = (V_G, E_G, L_G)$ be graphs.
- A function $h : V_P \rightarrow V_G$ is a homomorphism if it is edge preserving $(v, w) \in E_p \Rightarrow (h(v), h(w)) \in E_G$ and label preserving.



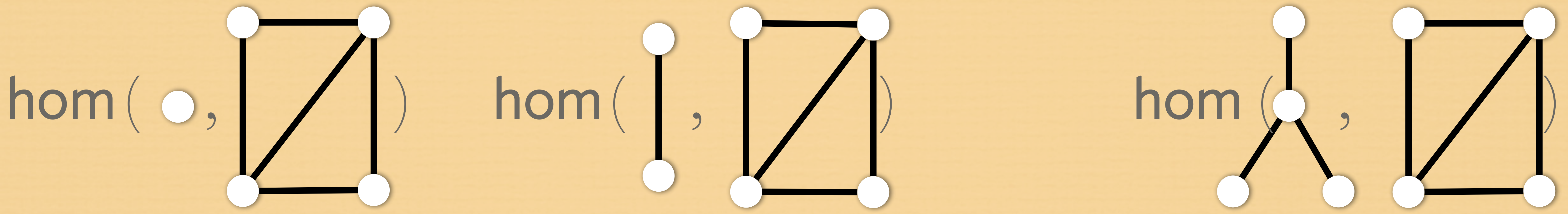
Homomorphism counts

- Define $\mathsf{HOM}(P, G) := \{$ all homomorphisms from P to $G\}$
- Define $\mathsf{hom}(P, G) := |\mathsf{HOM}(P, G)|$.



Homomorphism counts

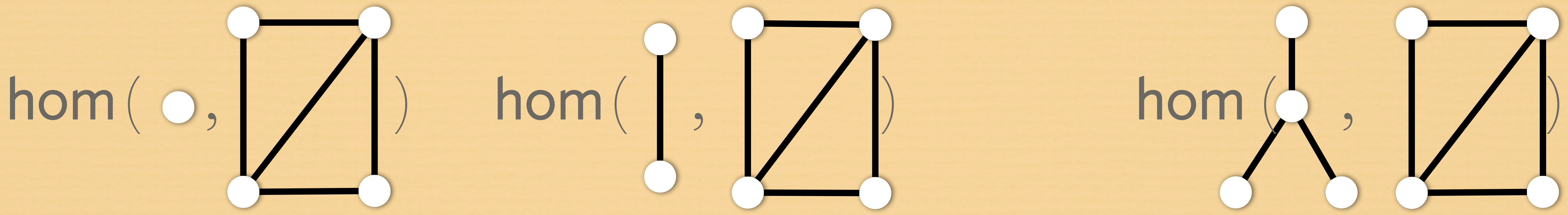
- Define $\mathrm{HOM}(P, G)$:= { all homomorphisms from P to G}
- Define $\mathrm{hom}(P, G)$:= $|\mathrm{HOM}(P, G)|$.



$\#\text{vertices} = 4$

Homomorphism counts

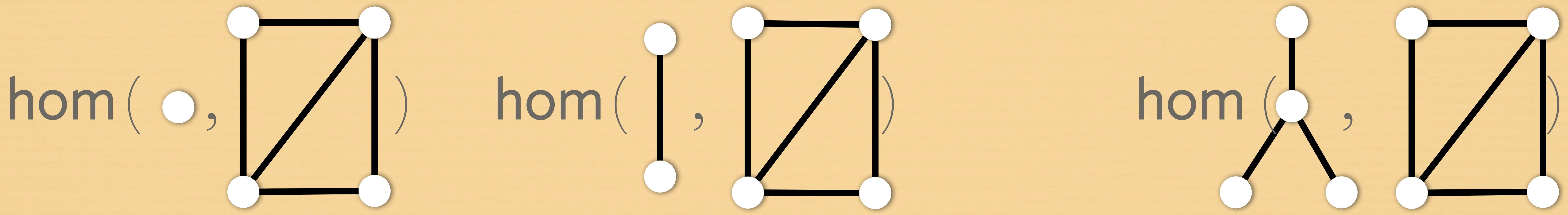
- Define $\mathsf{HOM}(P, G)$:= { all homomorphisms from P to G}
- Define $\mathsf{hom}(P, G)$:= $|\mathsf{HOM}(P, G)|$.



#vertices = 4 2#edges=10

Homomorphism counts

- Define $\mathsf{HOM}(P, G) := \{$ all homomorphisms from P to $G\}$
- Define $\mathsf{hom}(P, G) := |\mathsf{HOM}(P, G)|$.



$\#\text{vertices} = 4$ $\quad$ $2\#\text{edges}=10$ $\quad$ $70 = 2 \cdot 2^3 + 2 \cdot 3^3$

Graph isomorphisms and homomorphisms

Theorem (Lovász 1967)

Two graph G and H are isomorphic if and only if

$$\mathsf{hom}(P, G) = \mathsf{hom}(P, H)$$

for all graphs P

Graph isomorphisms and homomorphisms

Theorem (Lovász 1967)

Two graph G and H are isomorphic if and only if
$$\mathsf{hom}(P, G) = \mathsf{hom}(P, H)$$
for all graphs P

Homomorphism count information is very insightful and a complete invariant.

Graph isomorphisms and homomorphisms

Theorem (Lovász 1967)

Two graph G and H are isomorphic if and only if
$$\mathsf{hom}(P, G) = \mathsf{hom}(P, H)$$
for all graphs P

Homomorphism count information is very insightful and a complete invariant.

What if we do not consider all graphs P?

Homomorphism count vectors

- We can consider $\rho(\mathscr{P})$ for a set of graphs $\mathscr{P}$:

$$\rho(\mathscr{P}) := \{(G,H) \mid \forall P \in \mathscr{P} : \mathsf{hom}(P,G) = \mathsf{hom}(P,H)\}$$

Theorem (Lovász 1967)

Two graph G and H are isomorphic if and only if $(G,H) \in \rho(\mathscr{P})$ for $\mathscr{P}$ the set of all graphs.

Beautiful characterisations

Theorem (Dell et al. 2019, Dvorák 2010)

$$(G, H) \in \rho(\mathsf{C}_2)$$
if and only if
$$\mathsf{hom}(T, G) = \mathsf{hom}(T, H) \text{ for all } \textit{trees } T$$
if and only if
$$(G, H) \in \rho(\mathcal{T}) \textit{ for } \mathcal{T} \textit{ the set of all trees.}$$

Important class of MPNNs can only detect tree-based information

Z. Dvoräk: *On recognizing graphs by numbers of homomorphisms.* (2010)
Dell, Grohe, Rattan: *Lovász meets Weisfeiler and Leman.* (2018)
Cai, Fürer, Immerman: *An optimal lower bound on the number of variables for graph identification.* (1992)
M. Grohe: *The logic of graph neural networks.* (2021)

Beautiful characterisations

Theorem (Dell et al. 2019, Dvorák 2010)

$$(G, H) \in \rho(\mathsf{C}_2)$$

if and only if

$$\mathrm{hom}(T, G) = \mathrm{hom}(T, H) \text{ for all } \textit{trees } T$$

if and only if

$$(G, H) \in \rho(\mathcal{T}) \textit{ for } \mathcal{T} \textit{ the set of all trees.}$$

Important class of MPNNs can only detect tree-based information

All embedding methods in $\mathrm{GEL}_2(\Omega, \Theta)$ can only distinguish graphs based on **tree information!**

Z. Dvoräk: *On recognizing graphs by numbers of homomorphisms.* (2010)
Dell, Grohe, Rattan: *Lovász meets Weisfeiler and Leman.* (2018)
Cai, Fürer, Immerman: *An optimal lower bound on the number of variables for graph identification.* (1992)
M. Grohe: *The logic of graph neural networks.* (2021)

Beautiful characterisations

Theorem (Dell et al. 2019, Dvorák 2010)

$$(G, H) \in \rho(\mathsf{C}_{k+1})$$

if and only if

$$\mathsf{hom}(P, G) = \mathsf{hom}(P, H) \text{ for all } \textit{graphs } P \textit{ of treewidth } k$$

if and only if

$$(G, H) \in \rho(\mathcal{T}_k) \textit{ for } \mathcal{T}_k \textit{ the set of all graphs of tree width } k$$

Z. Dvorák: *On recognizing graphs by numbers of homomorphisms.* (2010)
Dell, Grohe, Rattan: *Lovász meets Weisfeiler and Leman.* (2018)
Cai, Fürer, Immerman: *An optimal lower bound on the number of variables for graph identification.* (1992)
M. Grohe: *The logic of graph neural networks.* (2021)

Beautiful characterisations

Theorem (Dell et al. 2019, Dvorák 2010)

$$(G,H) \in \rho(\mathsf{C}_{k+1})$$

if and only if

$\mathsf{hom}(P,G) = \mathsf{hom}(P,H)$ for all *graphs* P *of treewidth* k

if and only if

$(G,H) \in \rho(\mathcal{T}_k)$ *for* $\mathcal{T}_k$ *the set of all graphs of tree width* k

All embedding methods in $\mathrm{GEL}_k(\Omega,\Theta)$ can only distinguish graphs based on **treewidth** $k-1$ **pattern information!**

Z. Dvoräk: *On recognizing graphs by numbers of homomorphisms.* (2010)
Dell, Grohe, Rattan: *Lovász meets Weisfeiler and Leman.* (2018)
Cai, Fürer, Immerman: *An optimal lower bound on the number of variables for graph identification.* (1992)
M. Grohe: *The logic of graph neural networks.* (2021)

Treewidth

- A k-tree is a graph that can be obtained starting from a (k+1)-clique and then iteratively adding a vertex connected to a k-clique

k=2

Treewidth

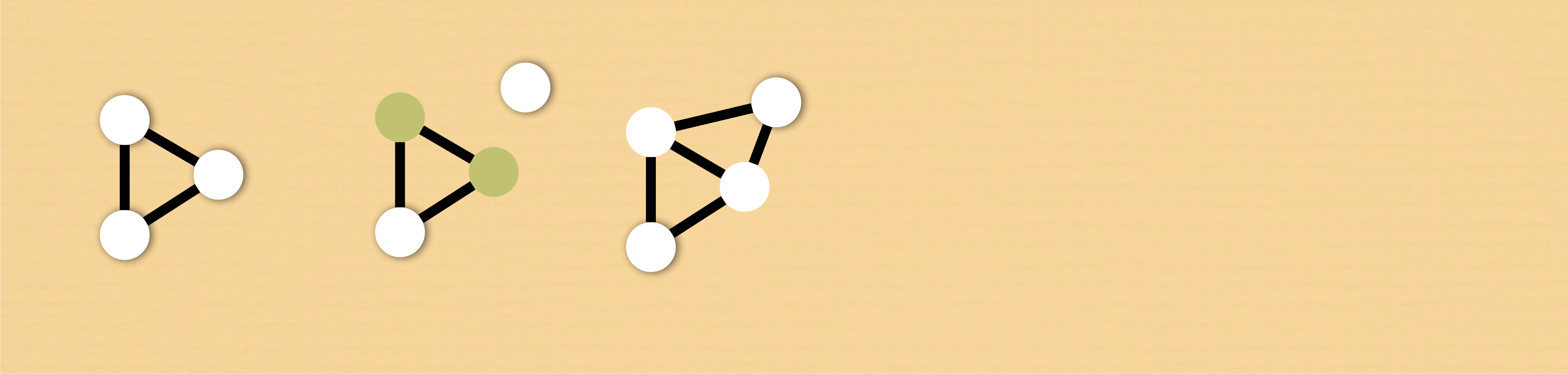
- A k-tree is a graph that can be obtained starting from a (k+1)-clique and then iteratively adding a vertex connected to a k-clique

k=2

Treewidth

- A k-tree is a graph that can be obtained starting from a (k+1)-clique and then iteratively adding a vertex connected to a k-clique

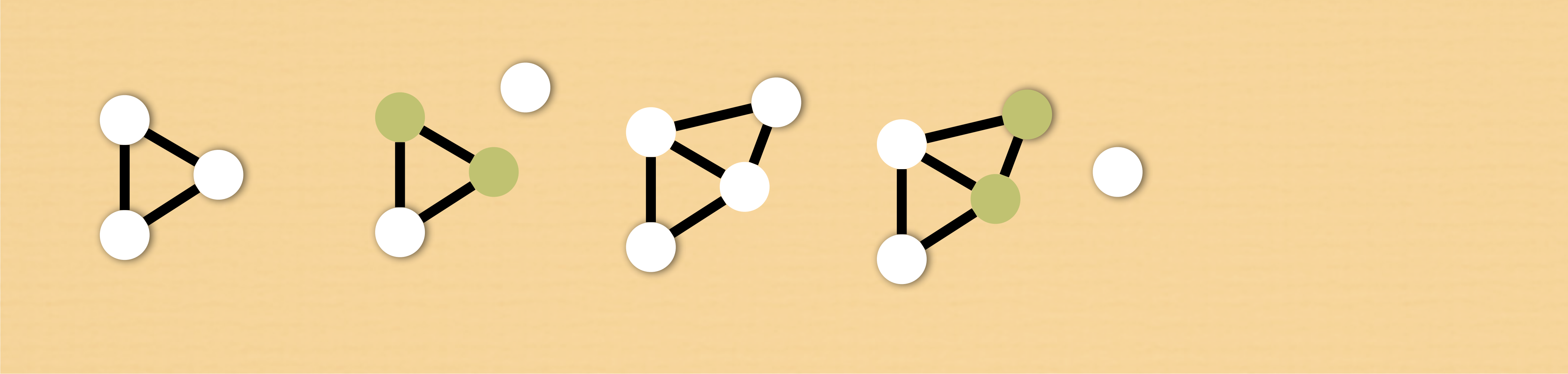
k=2



Treewidth

- A k-tree is a graph that can be obtained starting from a (k+1)-clique and then iteratively adding a vertex connected to a k-clique

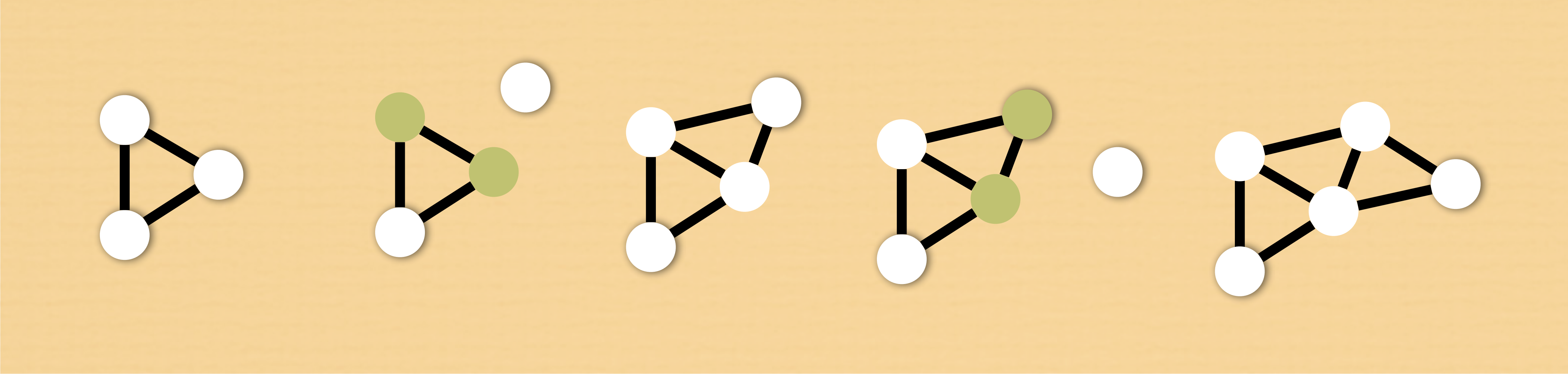
k=2



Treewidth

- A k-tree is a graph that can be obtained starting from a (k+1)-clique and then iteratively adding a vertex connected to a k-clique

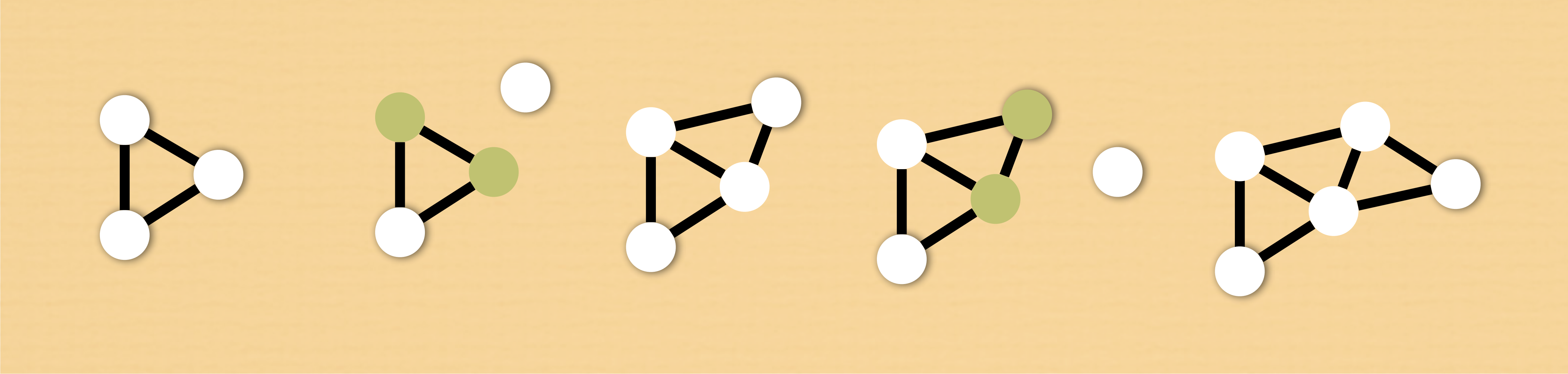
k=2



Treewidth

- A k-tree is a graph that can be obtained starting from a (k+1)-clique and then iteratively adding a vertex connected to a k-clique

k=2

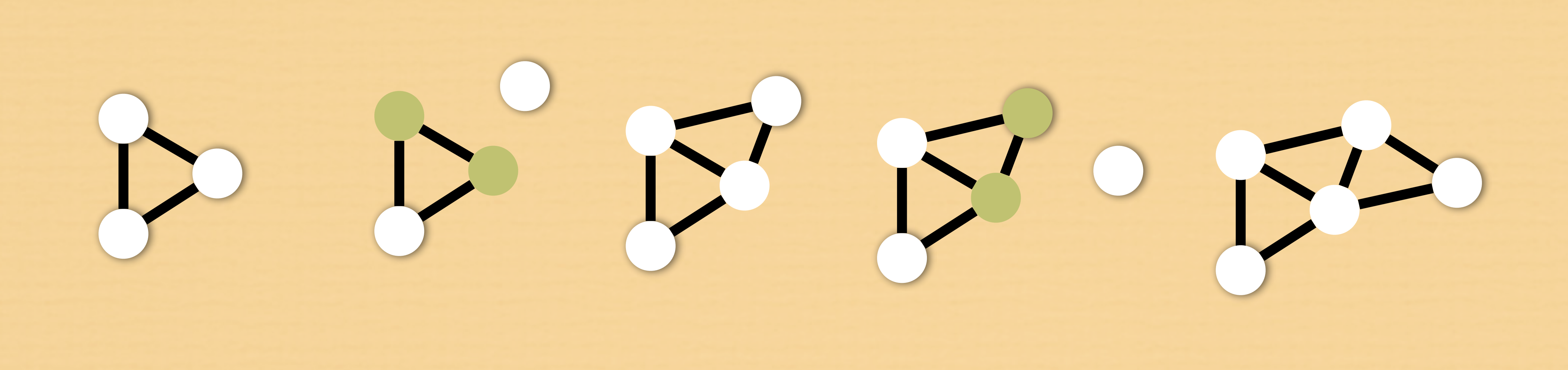


- A partial k-tree is a subgraph of a k-tree

Treewidth

- A k-tree is a graph that can be obtained starting from a (k+1)-clique and then iteratively adding a vertex connected to a k-clique

k=2



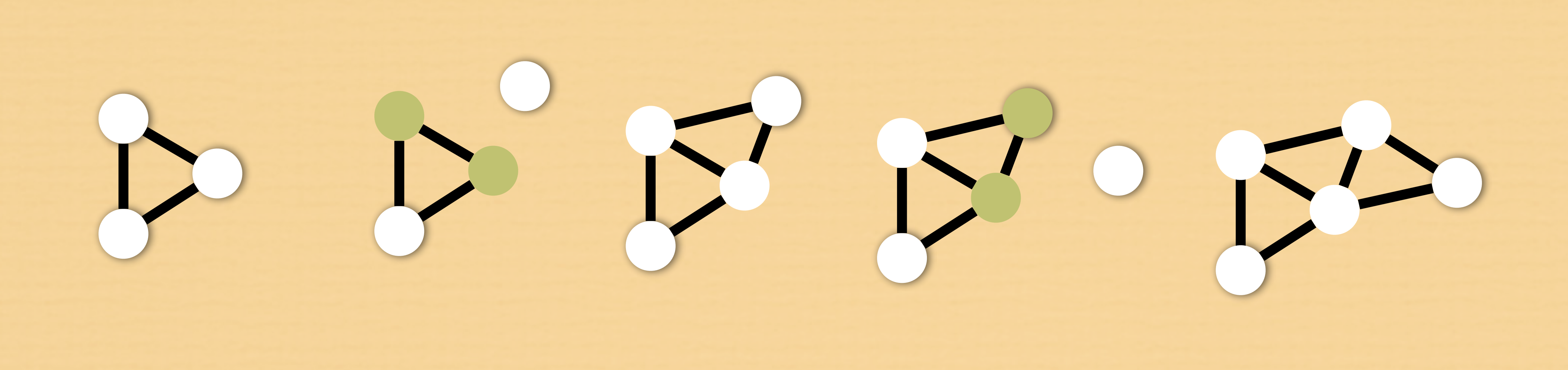
- A partial k-tree is a subgraph of a k-tree

Treewidth of a graph is smallest k such that the graph is a partial k-tree

Treewidth

- A k-tree is a graph that can be obtained starting from a (k+1)-clique and then iteratively adding a vertex connected to a k-clique

k=2



- A partial k-tree is a subgraph of a k-tree

Treewidth of a graph is smallest k such that the graph is a partial k-tree

- Trees=Treewidth 1

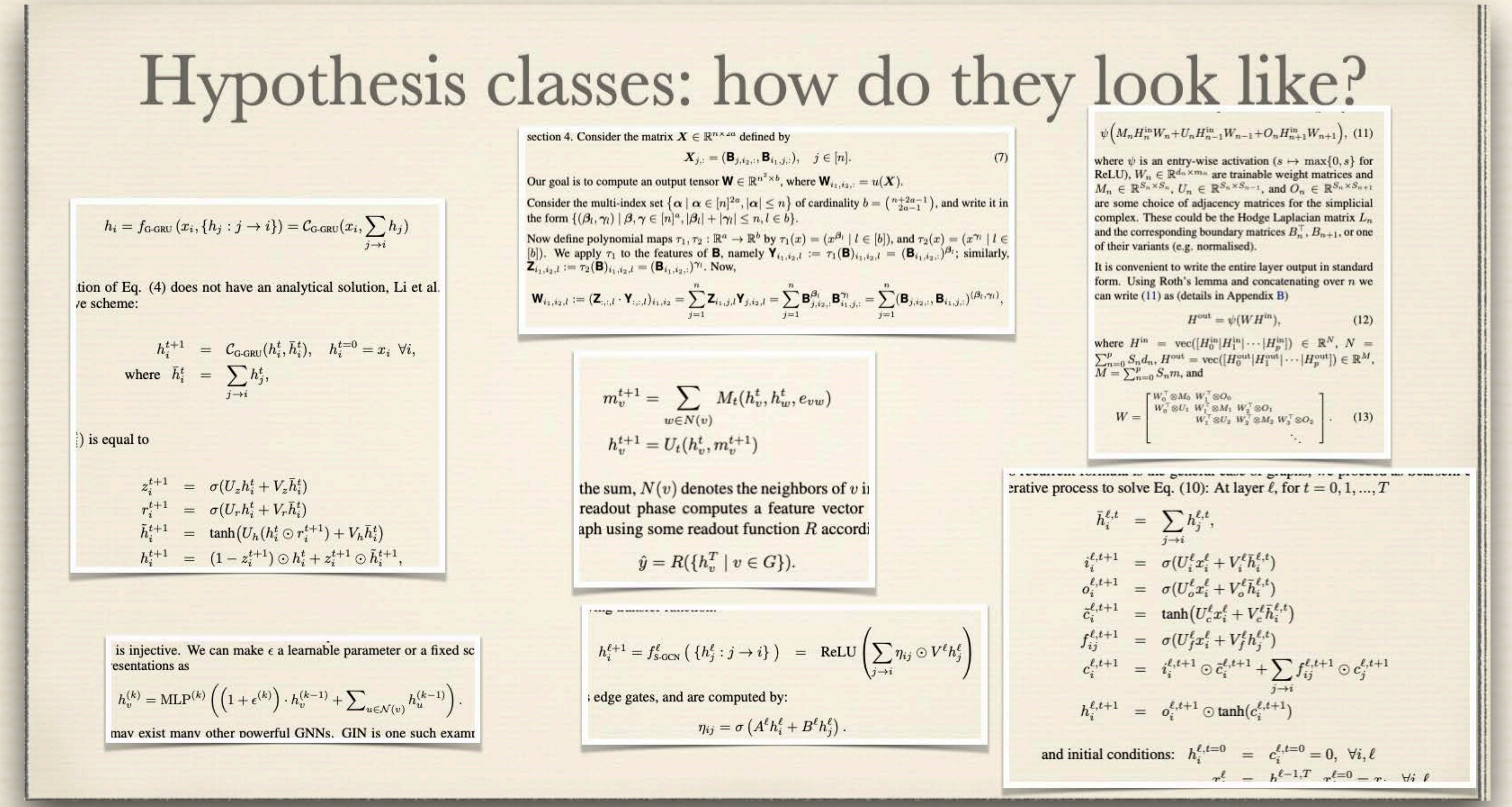
More connections

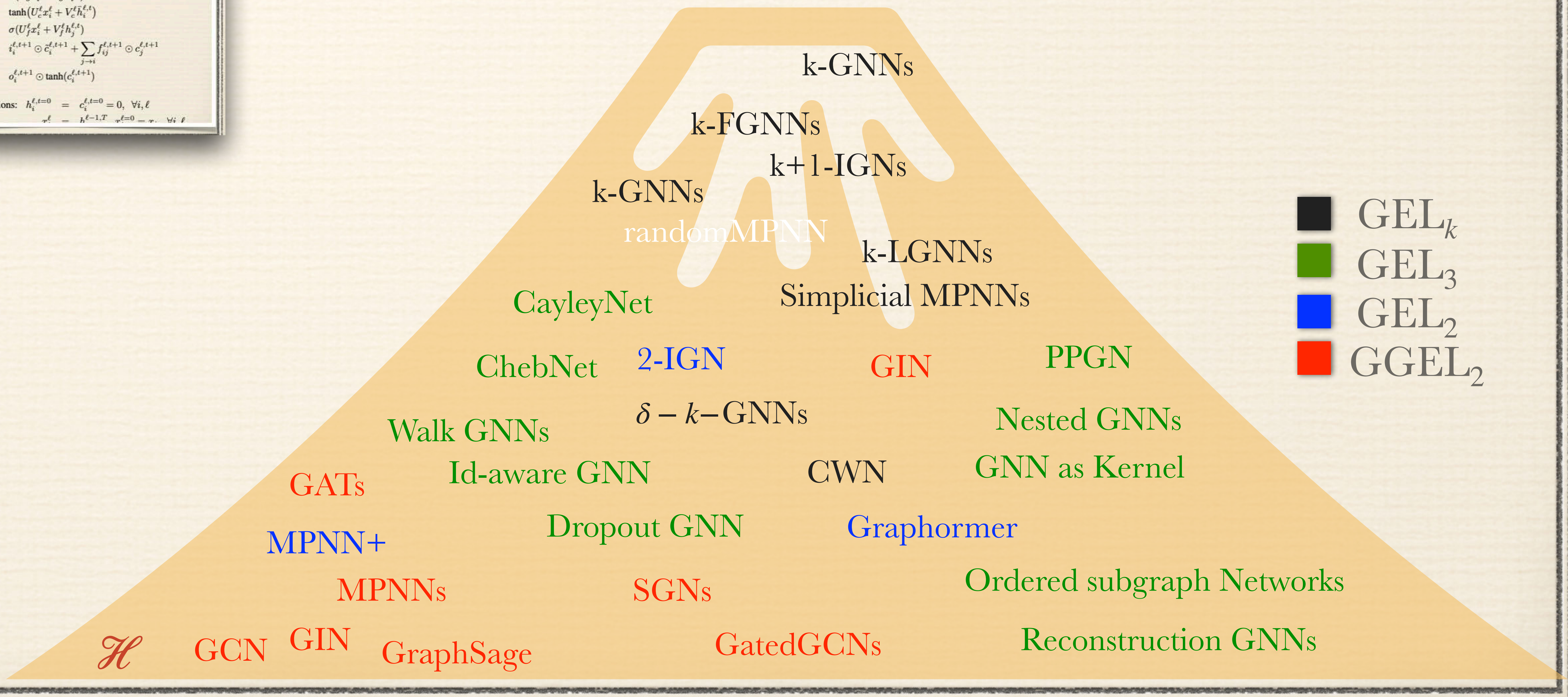
- To combinatorial graph algorithms color refinement and higher-dimensional Weisfeiler-Leman graph isomorphism tests.
- To linear algebraic congruences between adjacency matrices and systems of equations.
- To distance measures on graphs and metric equivalences.

Z. Dvorák: *On recognizing graphs by numbers of homomorphisms.* (2010)
Dell, Grohe, Rattan: *Lovász meets Weisfeiler and Leman.* (2018)
Cai, Fürer, Immerman: *An optimal lower bound on the number of variables for graph identification.* (1992)
M. Grohe: *The logic of graph neural networks.* (2021)

Takeaway message #2:

Classification in terms of logic, homomorphism counts, …





G., Reutter: *Expressiveness and approximation properties of graph neural networks.* (2022)

Expressive power

- Which inputs can be separated/distinguished by embeddings in $\mathscr{H}$.
- Which embeddings can be approximated by embeddings in $\mathscr{H}$?

$\mathscr{H}$ = class of embedding methods

Approximation properties

- Equip the set of graphs $\mathscr{G}$ with a topology and assume that $\mathscr{H}$ consists of continuous graph embeddings from $\mathscr{G}$ to $\mathbb{R}$.

- Let $\mathscr{C} \subseteq \mathscr{G}$ be a compact set of graphs.

Azizian, Lelarge: *Characterizing the expressive power of invariant and equivariant graph neural networks* (2021)
G., Reutter: *Expressiveness and approximation properties of graph neural networks* (2022)

Approximation properties

- Equip the set of graphs $\mathscr{G}$ with a topology and assume that $\mathscr{H}$ consists of continuous graph embeddings from $\mathscr{G}$ to $\mathbb{R}$.

- Let $\mathscr{C} \subseteq \mathscr{G}$ be a compact set of graphs.

Stone-Weierstrass

Theorem (Azizian & Lelarge 2021, G. and Reutter 2022)

If $\mathscr{H}$ is closed under linear combinations and product, then $\mathscr{H}$ can approximate any continuous function $\Xi : \mathscr{C} \to \mathbb{R}$ satisfying

$$\rho(\mathscr{H}) \subseteq \rho(\{\Xi\}).$$

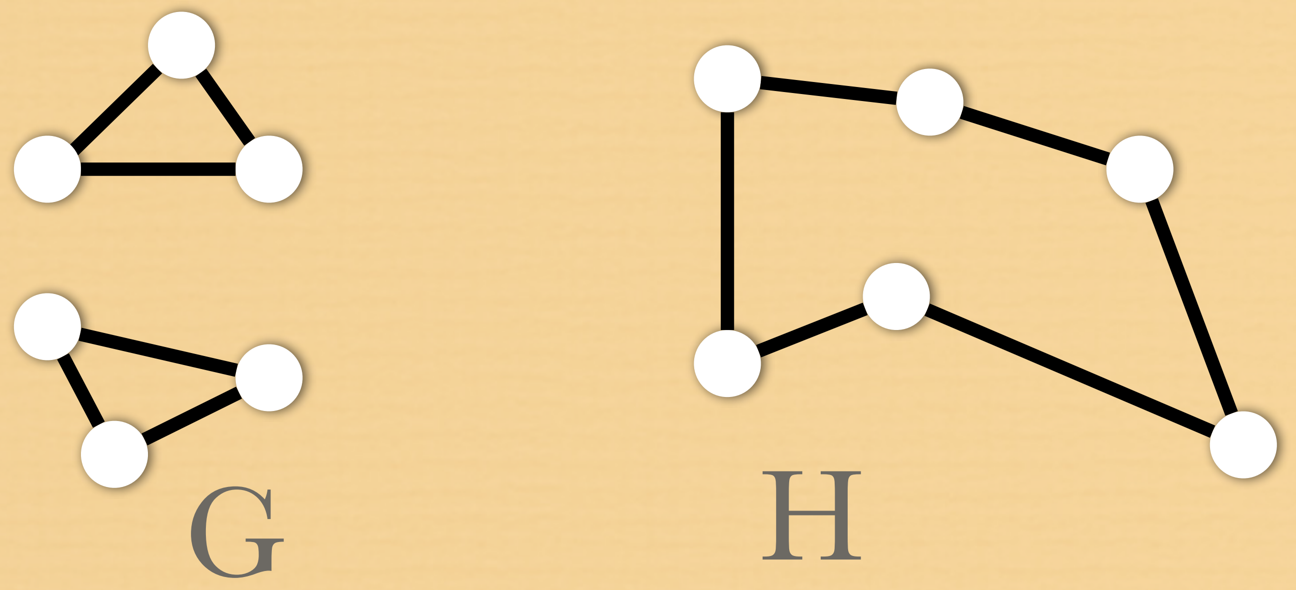
- Can be generalised to embeddings with output space $\mathbb{R}^d$

Azizian, Lelarge: *Characterizing the expressive power of invariant and equivariant graph neural networks* (2021)
G., Reutter: *Expressiveness and approximation properties of graph neural networks* (2022)

MPNNs: Approximation

Theorem

On compact set of graphs, MPNNs can approximate any continuous graph embedding $\Xi : \mathscr{C} \to \mathbb{R}$ satisfying $\rho(\mathsf{C}_2) \subseteq \rho(\{\Xi\})$



$(G, H) \in \rho(\mathrm{MPNN}) \Rightarrow$ Cannot approximate graph functions based on

- connected components
- 3-cliques
-

❖ Intricate relation between distinguishing power and approximation properties

Azizian, Lelarge: *Characterizing the expressive power of invariant and equivariant graph neural networks* (2021)
G., Reutter: *Expressiveness and approximation properties of graph neural networks* (2022)

Expressive power

- Which inputs can be separated/distinguished by embeddings in $\mathscr{H}$.
- Which embeddings can be approximated by embeddings in $\mathscr{H}$?
- What is the VC dimension of $\mathscr{H}$?

$\mathscr{H}$ = class of embedding methods

VC dimension

- A set of graphs $G_1, \ldots, G_s$ can be shattered by $\mathscr{H}$ if for any boolean vector $\tau \in \{0,1\}^s$, there is a $\xi_\tau \in \mathscr{H}$ such that $\xi_\tau(G_i) = \tau_i$ for all $i = 1,\ldots, s$
- We define the VC dimension of $\mathscr{H}$ on $\mathscr{G}' \subseteq \mathscr{G}$ as

$$\mathsf{VC}_{\mathscr{G}'}(\mathscr{H}) := \max\{s \mid \exists G_1, \ldots, G_s \text{ in } \mathscr{G}' \text{ which can be shattered by } \mathscr{H}\}$$

Theorem (Morris et al. 2023)

$$\mathsf{VC}_{\mathscr{G}'}(\mathscr{H}) \leq |\mathscr{G}'/_{\rho(\mathscr{H})}|$$

for some hypothesis classes also equality holds.

Equivalence classes induced by $\rho(\mathscr{H})$

Morris, G.,Tönshoff, Grohe; *WL meet VC* (2023).

Expressive power

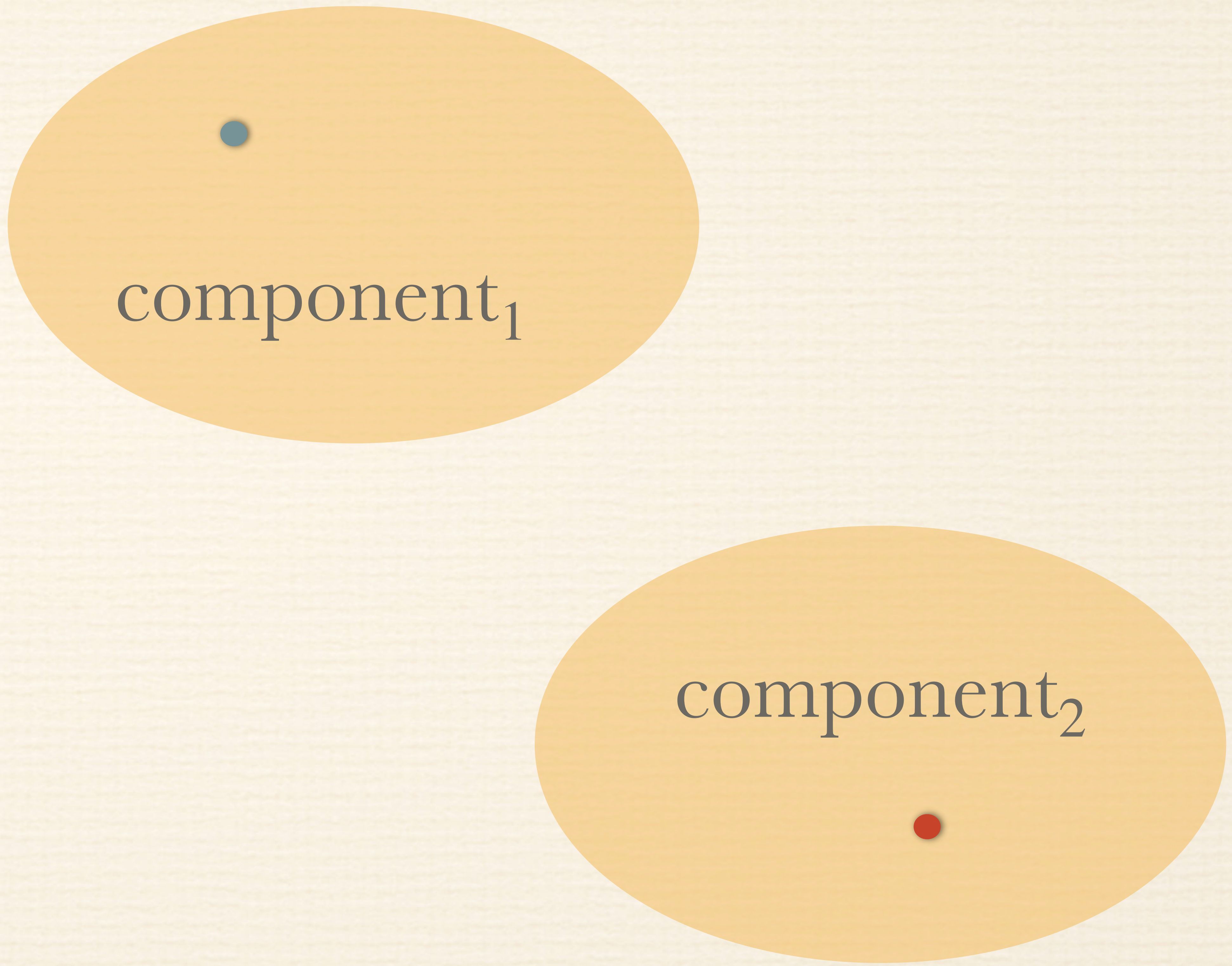
- Which inputs can be separated/distinguished by embeddings in $\mathscr{H}$.
- Which embeddings can be approximated by embeddings in $\mathscr{H}$?
- What is the VC dimension of $\mathscr{H}$?
- Which embeddings can be expressed by embeddings in $\mathscr{H}$?

$\mathscr{H}$ = class of embedding methods

Which unary C_2 formulas can MPNNs express?

- Not all: $\varphi(x) := L_b(x) \wedge \exists y L_r(y)$

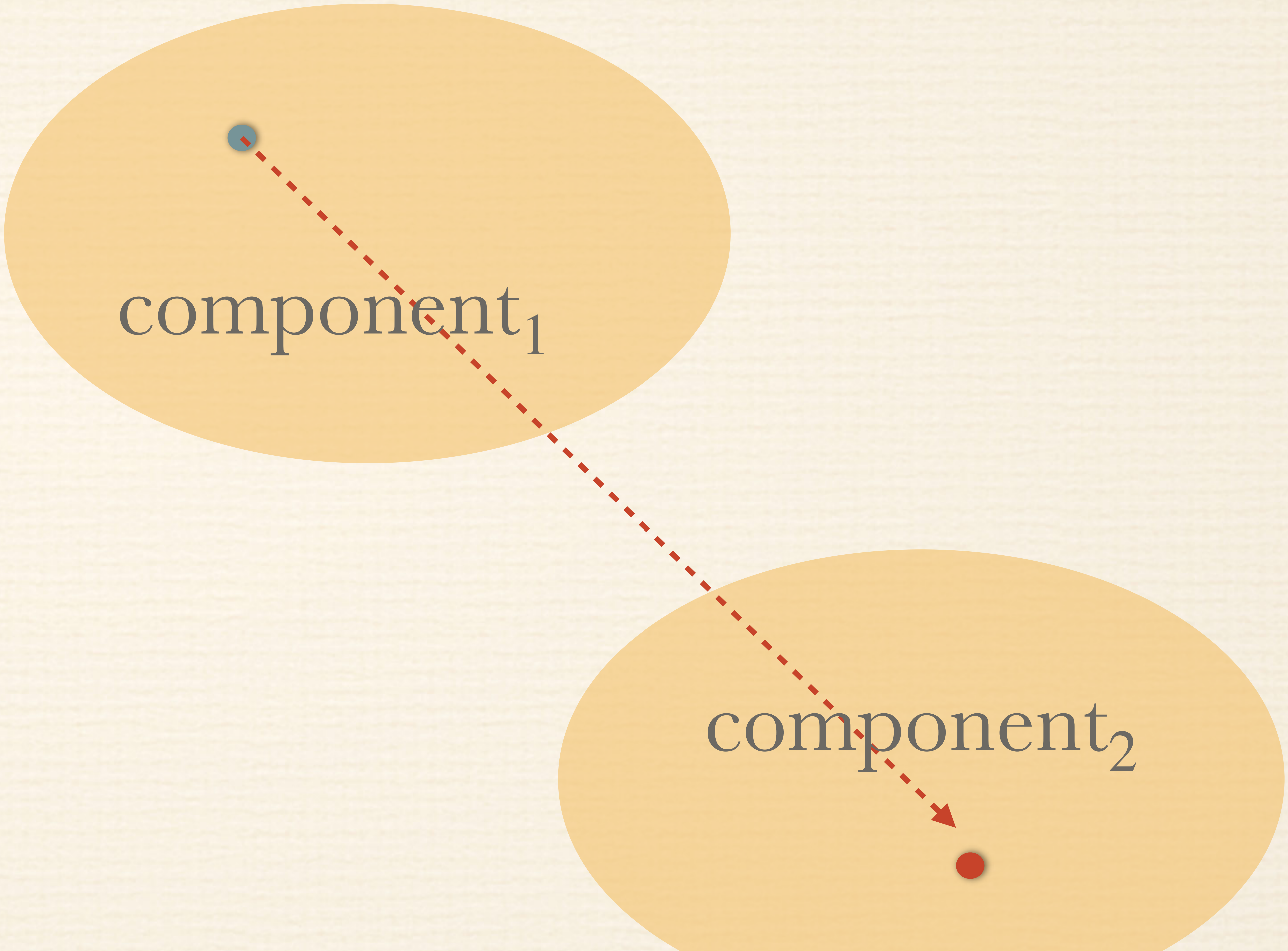
  I am blue and there exist
  a red vertex somewhere…



Which unary C_2 formulas can MPNNs express?

- Not all: $\varphi(x) := L_b(x) \wedge \exists y L_r(y)$

  I am blue and there exist
  a red vertex somewhere…



Cannot be reached by neighborhood aggregation

Which unary C_2 formulas can MPNNs express?

Theorem (Barceló et al. 2020)

Let $\varphi(x)$ be a unary C_2 formula. Then, $\varphi(x)$ is equivalent to a GC_2 formula *if and only if* $\varphi(x)$ is expressible by the class of MPNNs.

$$\exists \xi \in \text{MPNN} : \forall G \in \mathcal{G}, \forall v \in V_G : (G, v) \vDash \varphi \Leftrightarrow \xi(G, v) = 1$$

Barceló, Kostylev, Monet, Pérez, Reutter, Silva: *The logical expressiveness of graph neural networks* (2020)
Barceló, Kostylev, Monet, Pérez, Reutter, Silva: *The Expressive Power of Graph Neural Networks as a Query Language.* (2020)

MPNNs+

Allow for aggregation over all vertices not only edge-guarded

Theorem (Barceló et al. 2020)

Every unary C_2 formula $\varphi(x)$ is expressible by the class of MPNNs+

Of course, there are queries beyond C_2 which MPNNs can express

Barceló, Kostylev, Monet, Pérez, Reutter, Silva: *The logical expressiveness of graph neural networks* (2020)
Barceló, Kostylev, Monet, Pérez, Reutter, Silva: *The Expressive Power of Graph Neural Networks as a Query Language.* (2020)

Descriptive complexity of GNNs

Theorem (Grohe 2023)

If a unary query Q is computable by a GNN with rational weights and piecewise linear activation functions, then Q is definable in the guarded fragment of $\mathsf{FO}_2 + \mathsf{C}$

Different from C_2
Two sorted logic, numerical predicates etc.

- Extends to general GNNs with real weights and more complex activation functions $\Rightarrow$ approximate with GNNs as in theorem

M. Grohe. *The Descriptive Complexity of Graph Neural Networks* (2023)

Descriptive complexity of GNNs

Theorem (Grohe 2023)

If a unary query Q is computable by a GNN with rational weights and piecewise linear activation functions, then Q is definable in the guarded fragment of $FO_2 + \mathsf{C}$

Different from C_2
Two sorted logic, numerical predicates etc.

- Extends to general GNNs with real weights and more complex activation functions ⇒ approximate with GNNs as in theorem
- Situates queries expressible by GNNs in (non-uniform) TC^0

Boolean functions computable by non-uniform polynomial-size bounded-depth family of circuits with threshold gates

M. Grohe. *The Descriptive Complexity of Graph Neural Networks* (2023)

Descriptive complexity of GNNs

Theorem (Grohe 2023)

If a unary query Q is computable by a GNN with rational weights and piecewise linear activation functions, then Q is definable in the guarded fragment of $FO_2 + \mathsf{C}$

Different from C_2
Two sorted logic, numerical predicates etc.

- Extends to general GNNs with real weights and more complex activation functions ⇒ approximate with GNNs as in theorem
- Situates queries expressible by GNNs in (non-uniform) TC^0

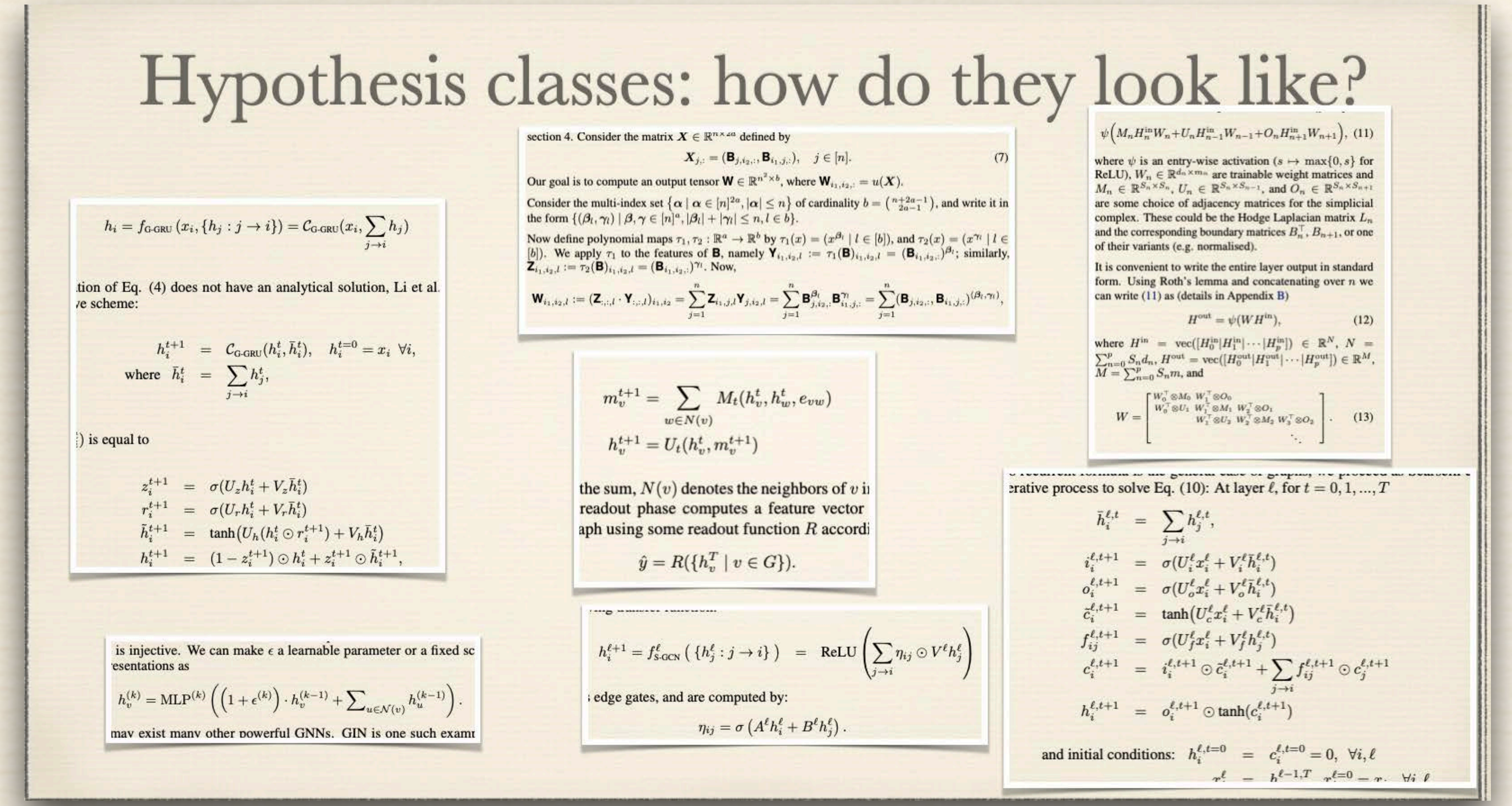
Boolean functions computable by non-uniform polynomial-size bounded-depth family of circuits with threshold gates

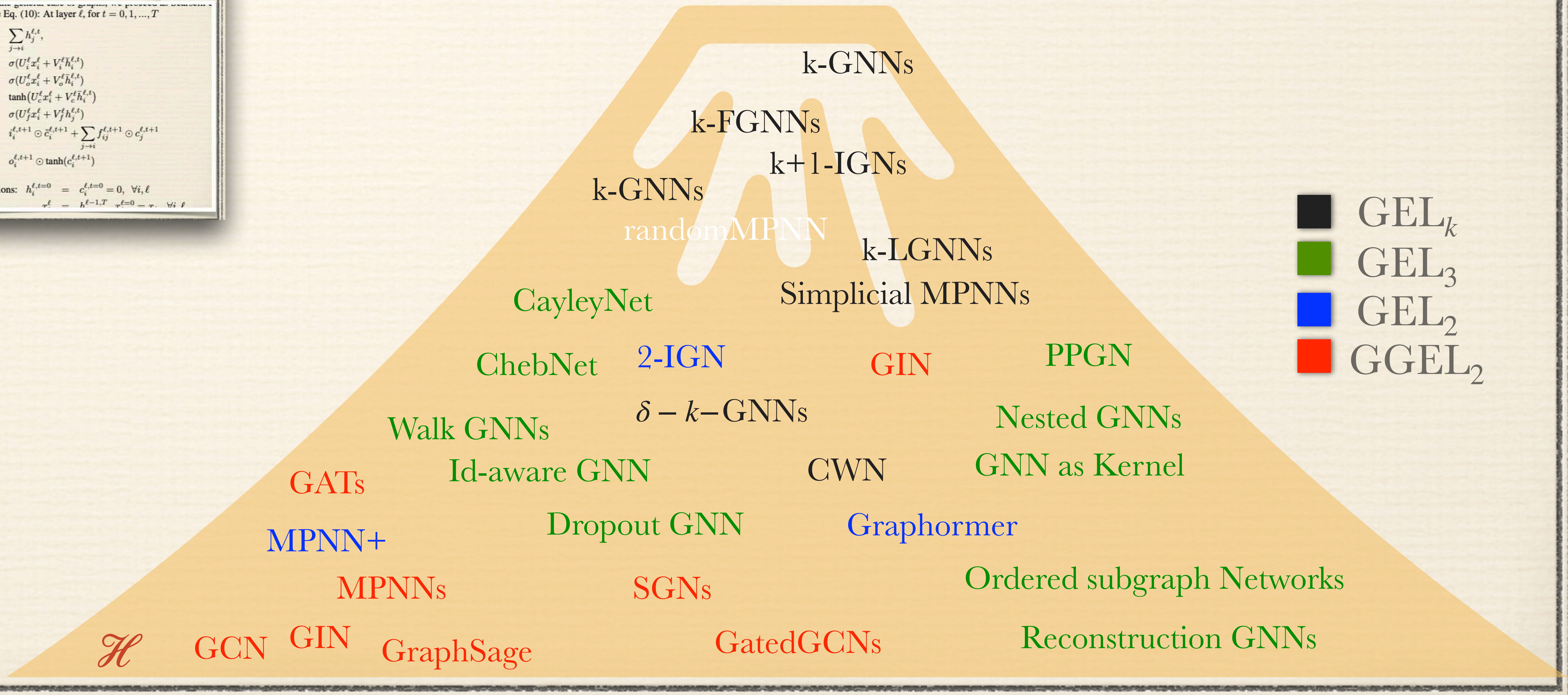
- Converse holds, with random vertex features.

M. Grohe. *The Descriptive Complexity of Graph Neural Networks* (2023)

Takeaway message #3:

Classification along different dimensions of expressibility, not only distinguishabilty





G., Reutter: *Expressiveness and approximation properties of graph neural networks.* (2022)

How to compare different classes?

- How to compare such embedding classes theoretically?
- How to bring order to the chaos?

1. See graph embedding methods as queries in some query lang

Distinguishability, approximation, generalisation, uniform and non-uniform expressiveness

2. Analyse expressive power of query language

3. Transfer understanding back to graph learning world

How to compare different classes?

- How to compare such embedding classes theoretically?
- How to bring order to the chaos?

GEL

1. See graph embedding methods as queries in some query lang

Distinguishability, approximation, generalisation, uniform and non-uniform expressiveness

2. Analyse expressive power of query language

3. Transfer understanding back to graph learning world

Conclusion

Todos

- MPNNs: Efficient, most widely used but not expressive (C_2)
- Methods matching C_k for $k > 2$ require tensors making them inefficient
- Ongoing efforts to boost power but preserve efficiency

Todos

- MPNNs: Efficient, most widely used but not expressive (C_2)
- Methods matching C_k for $k > 2$ require tensors making them inefficient
- Ongoing efforts to boost power but preserve efficiency

Feature augmentation

Precompute hom/iso counts

Bouritsas et al.: *Improving graph neural network expressivity via subgraph isomorphism counting* (2020)
Barceló et al.: *Graph neural networks with local graph parameters.* (2021)

Random features

Dasoulas et al.: *Coloring graph neural networks for node disambiguation* (2020)
Sato et al.: *Random features strengthen graph neural networks* (2021).
Abboud et al. :*The surprising power of graph neural networks with random node initialization.* (2021)

Spectral/Global properties

Kreuzer et al.: *Rethinking graph transformers by spectral attention* (2021)
Ying et al.: *Do transformers really perform bad for graph representation* (2021)
Lim et al.: *Sign and Basis Invariant Networks for Spectral Graph Representation Learning* (2022)
Zhang et al.: *Rethinking the expressive power of gnns via graph biconnectivity* (2023)] [2]

Todos

- MPNNs: Efficient, most widely used but not expressive (C_2)
- Methods matching C_k for $k > 2$ require tensors making them inefficient
- Ongoing efforts to boost power but preserve efficiency

Feature augmentation

Precompute hom/iso counts

Bouritsas et al.: *Improving graph neural network expressivity vi[...] graph isomorphism counting* (2020)
Barceló et al.: *Graph neural networks with local graph [...]*

Random features

Dasoulas et al.: *Coloring gra[...]*
Sato et al.: *Random featur[...]*
Abboud et al. :*The surpr[...]*

Spectral/Glo[...]

Kreuzer et al.: *Rethinking gra[...]*
Ying et al.: *Do transformers really p[...]*
Lim et al.: *Sign and Basis Invariant Netwo[...]*
Zhang et al.: *Rethinking the expressive power of gnns [...]*

Running graph learning method on a derived view.
Analysis of expressive power (logic, hom count,…)

Todos

- MPNNs: Efficient, most widely used but not expressive (C_2)
- Methods matching C_k for $k > 2$ require tensors making them inefficient
- Ongoing efforts to boost power but preserve efficiency

Feature augmentation

Precompute hom/iso counts

Bouritsas et al.: *Improving graph neural network expressivity vi*[...]*raph isomorphism counting* (2020)
Barceló et al.: *Graph neural networks with local graph*[...]

Random feature[...]

Dasoulas et al.: *Coloring gra*[...]
Sato et al.: *Random featur*[...]
Abboud et al. :*The surpr*[...]

Spectral/Glo[...]

Kreuzer et al.: *Rethinking gra*[...]
Ying et al.: *Do transformers really p*[...]
Lim et al.: *Sign and Basis Invariant Netwo*[...]
Zhang et al.: *Rethinking the expressive power of gnns via*[...]

Running graph learning
method on a derived view.
Analysis of expressive power (logic,
hom count,…)

Subgraph GNNs

Bevilacqua et al: *Equivariant subgraph aggregation network* (2022)
Cotta et al.: *Reconstruction for powerful graph representations* (2021)
Bevilacqua et al.: *Understanding and extending subgraph GNNs by rethinking their symmetries* (2022)
Huang et al.: *Boosting the cycle counting power of graph neural networks with I2-GNNs* (2022)
Papp et al.: *DropGNN: Random dropouts increase the expressiveness of graph neural networks.* (2021)
Qian et al.: *Ordered subgraph aggregation networks.* (2022)
You et al.: *Identity-aware graph neural networks.* (2021)
Zhang and P. Li. *Nested graph neural networks* (2021)
Zhao et al.: *From stars to subgraphs: Uplifting any GNN with local structure awareness* (2022)

Todos

- MPNNs: Efficient, most widely used but not expressive (C_2)
- Methods matching C_k for $k > 2$ require tensors making them inefficient
- Ongoing efforts to boost power but preserve efficiency

Feature augmentation

Precompute hom/iso counts

Bouritsas et al.: *Improving graph neural network expressivity vi… raph isomorphism counting* (2020)
Barceló et al.: *Graph neural networks with local graph…*

Random features

Dasoulas et al.: *Coloring gra…*
Sato et al.: *Random featur…*
Abboud et al. :*The surpr…*

Spectral/Glo…

Kreuzer et al.: *Rethinking gra…*
Ying et al.: *Do transformers really p…*
Lim et al.: *Sign and Basis Invariant Netwo…*
Zhang et al.: *Rethinking the expressive power of gnns via g…*

Running graph learning method on a derived view. Analysis of expressive power (logic, hom count,…)

Subgraph GNNs

Bevilacqua et al: *Equivariant subgraph aggregation network…* …22)
Cotta et al.: *Reconstruction for powerful graph representations* …21)
Bevilacqua et al.: *Understanding and extending subgraph G… rethinking their symmetries* (2022)
Huang et al.: *Boosting the cycle counting power of…* …(2022)
Papp et al.: *DropGNN: Random drop…* …21)
Qian et al.: *Ordered subgraph…*
You et al.: *Identity-aware…*
Zhang and P. Li. *Nest…*
Zhao et al.: *From sta…*

Running graph learning method on many views, then aggregate. Analysis of expressive power

Todos

- Number of variables depends on GEL skills, are there better notions?
- Specialized homomorphism count characterizations, more fine grained than logic?
- Analysis does not always explain experiments. Is a more fine grained analysis possible, perhaps taking learning process into account?
- Relational embedding methods.
- Recurrent GNNs are closely related to fixpoint computations. Relationship to query languages with recursion?

Recipe for upper bounding architectures

1. **Take you GNN** architecture and **write it in GEL**, but using *a minimal number of variables* of variables.

2. Call this *number* of variables **k**.

3. Then the *power* of your architecture is **bounded by** C_k