Graph Neural Networks: Asymptotics and Unbounded Expressivity

Advanced ML, Trento, October 2026

Manfred Jaeger

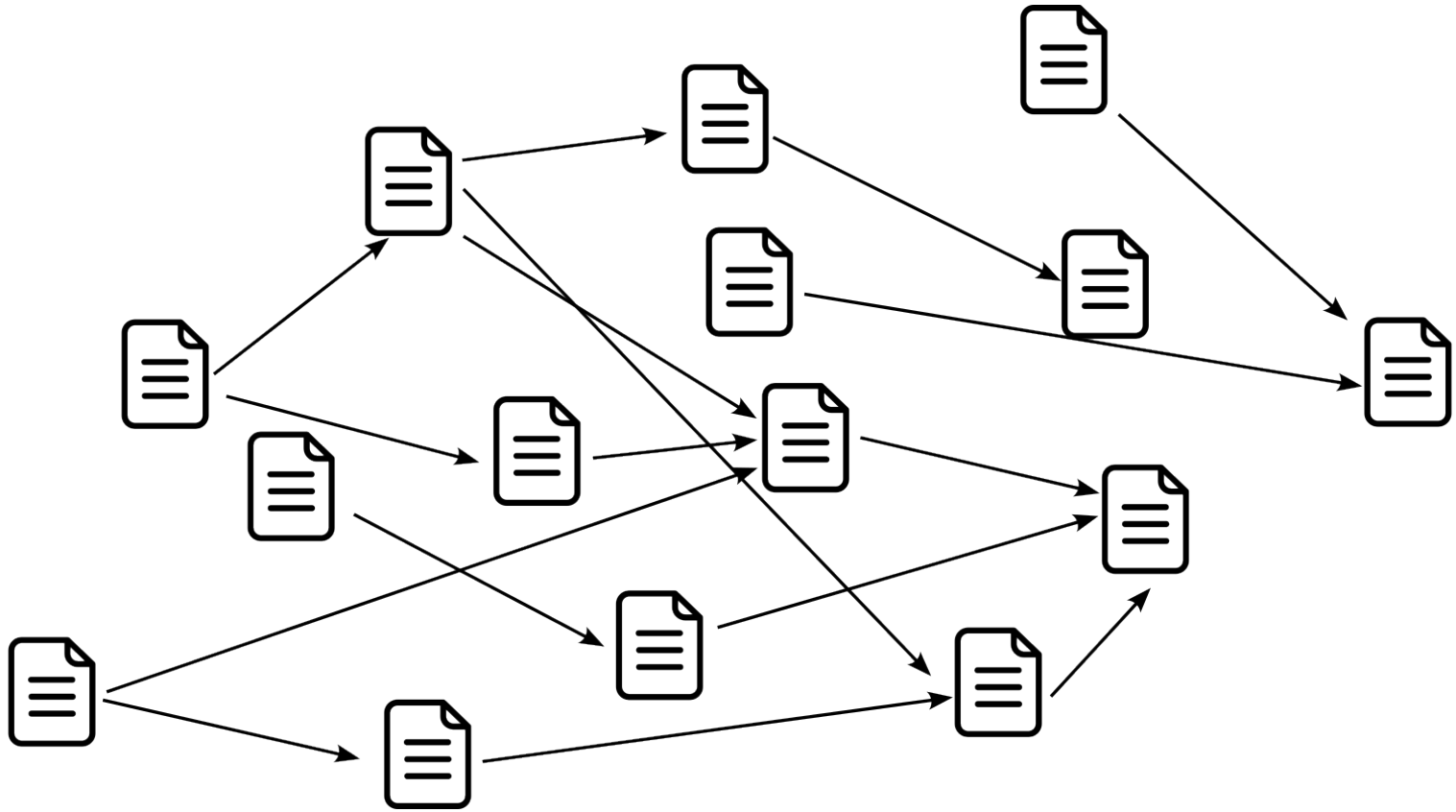


The plan for today:

- A quick reminder: graph neural networks (GNNs)
- Theoretical questions:
  - how expressive are different GNN architectures?
  - how robust are different GNN architectures w.r.t. changes (increase) in the input graph size?
  - what is the relationship between expressivity and robustness?
- Literature: some articles cited on the slides + unpublished material

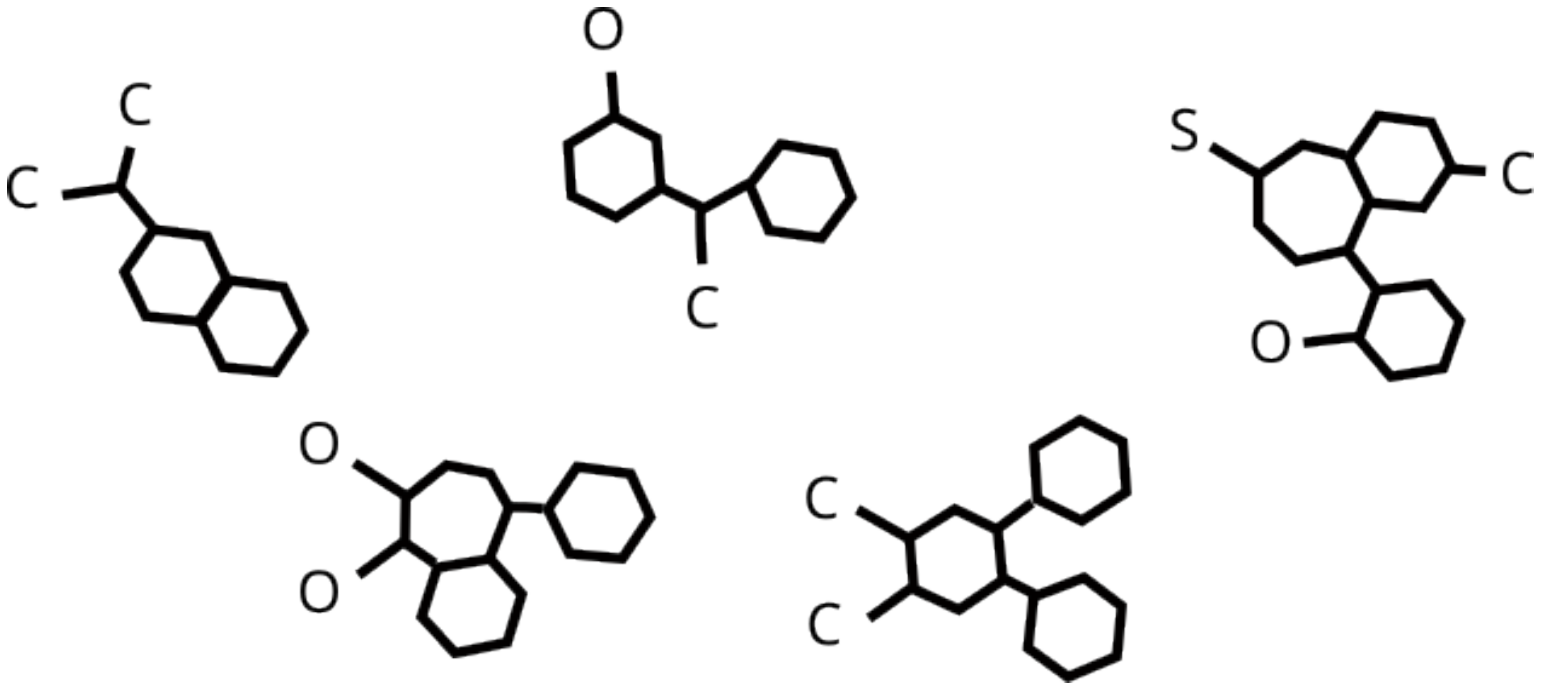
GNNs: A Reminder

A bibliographic graph:



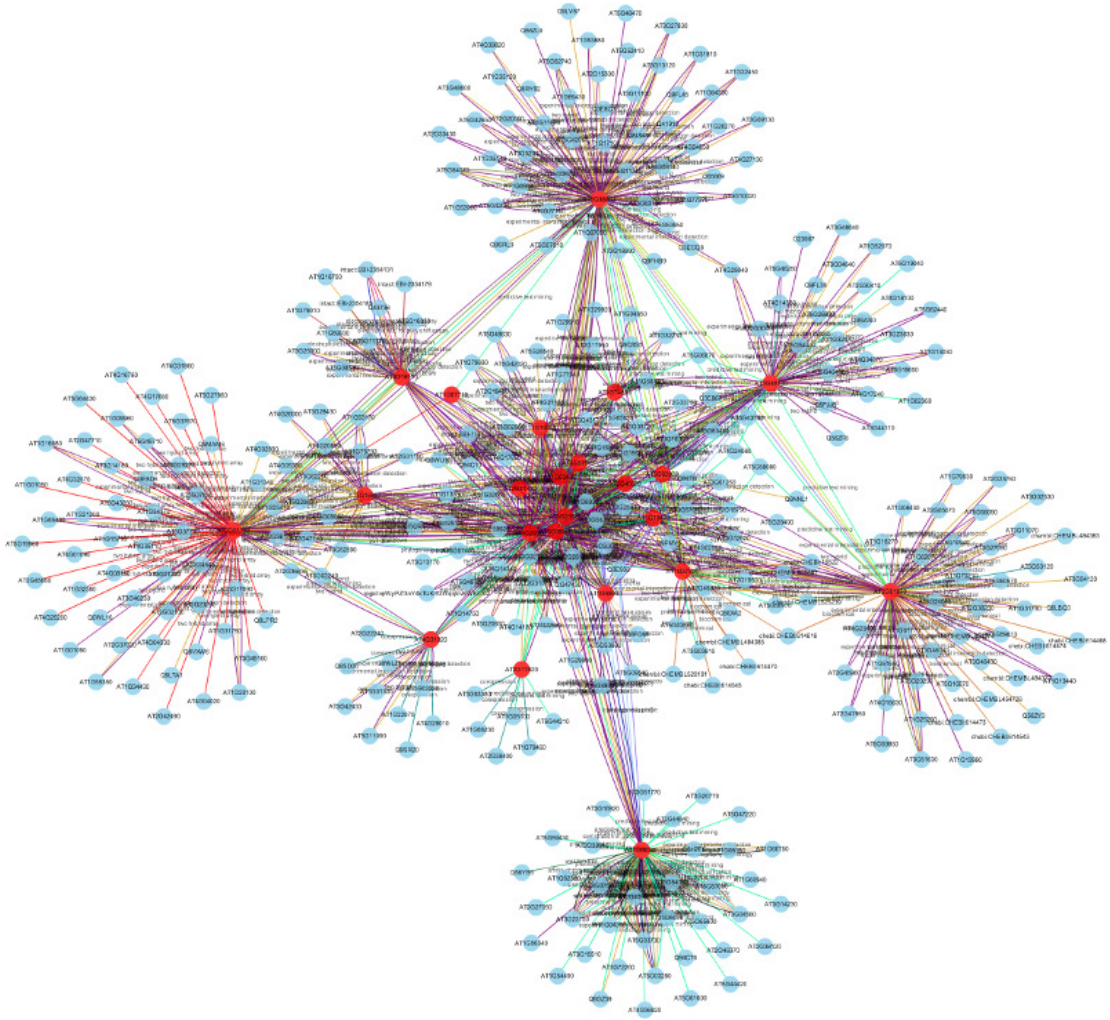
➥Predict the *topic* label of a research paper.

Molecules as graphs:

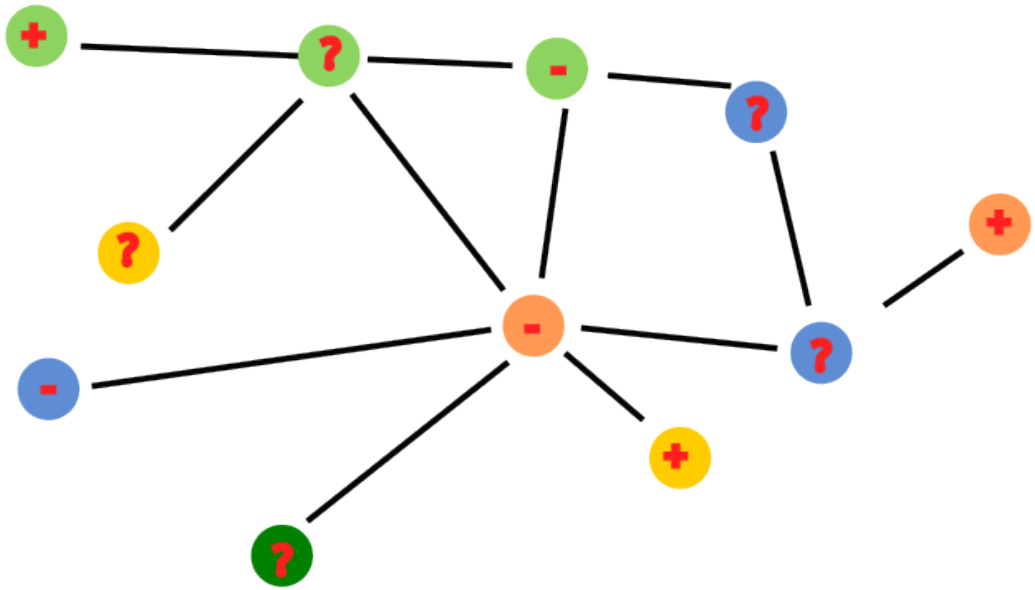


➥Predict the *toxic* label of a molecule.

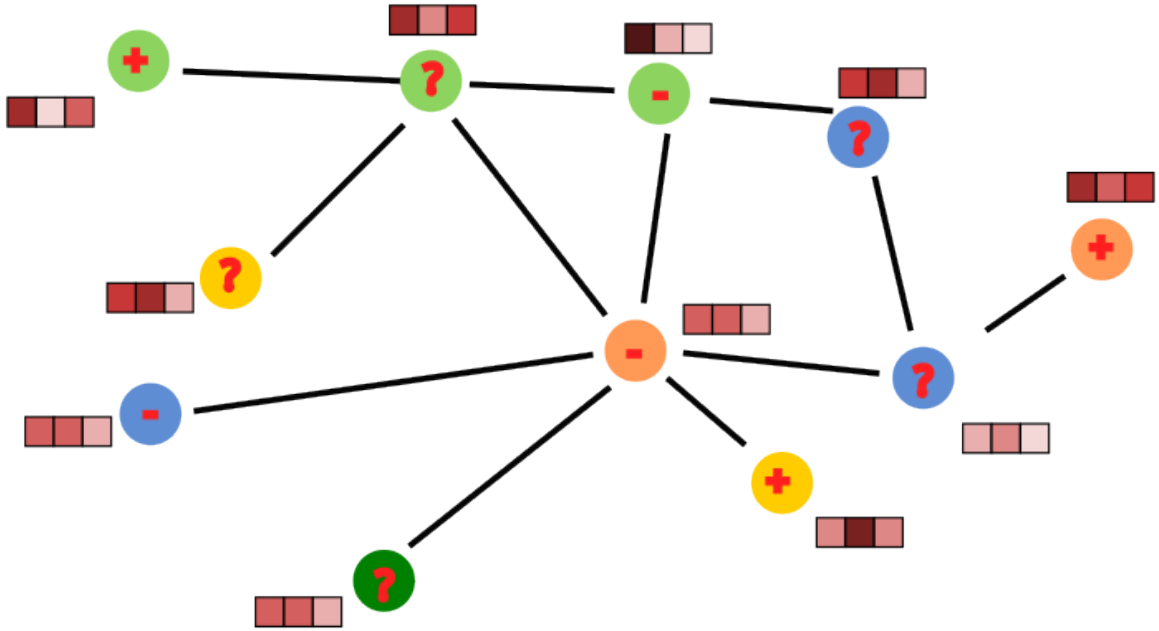
Protein-protein interaction network:



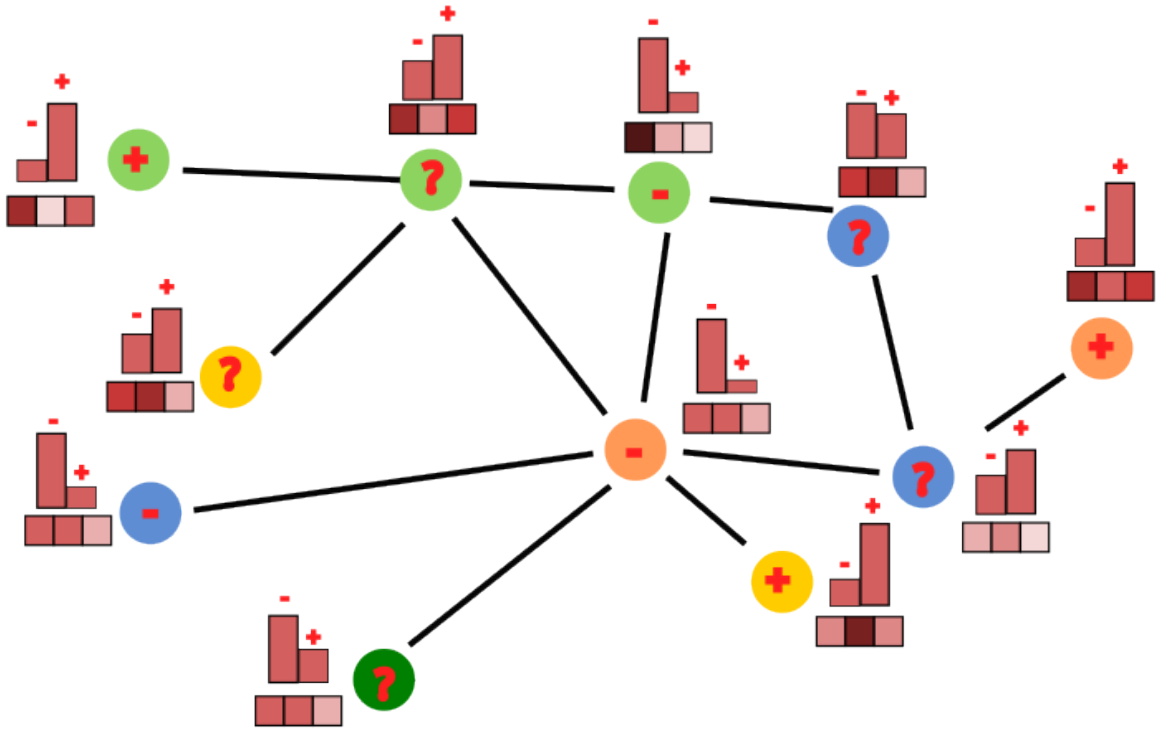
➥Predict missing *interaction* links.



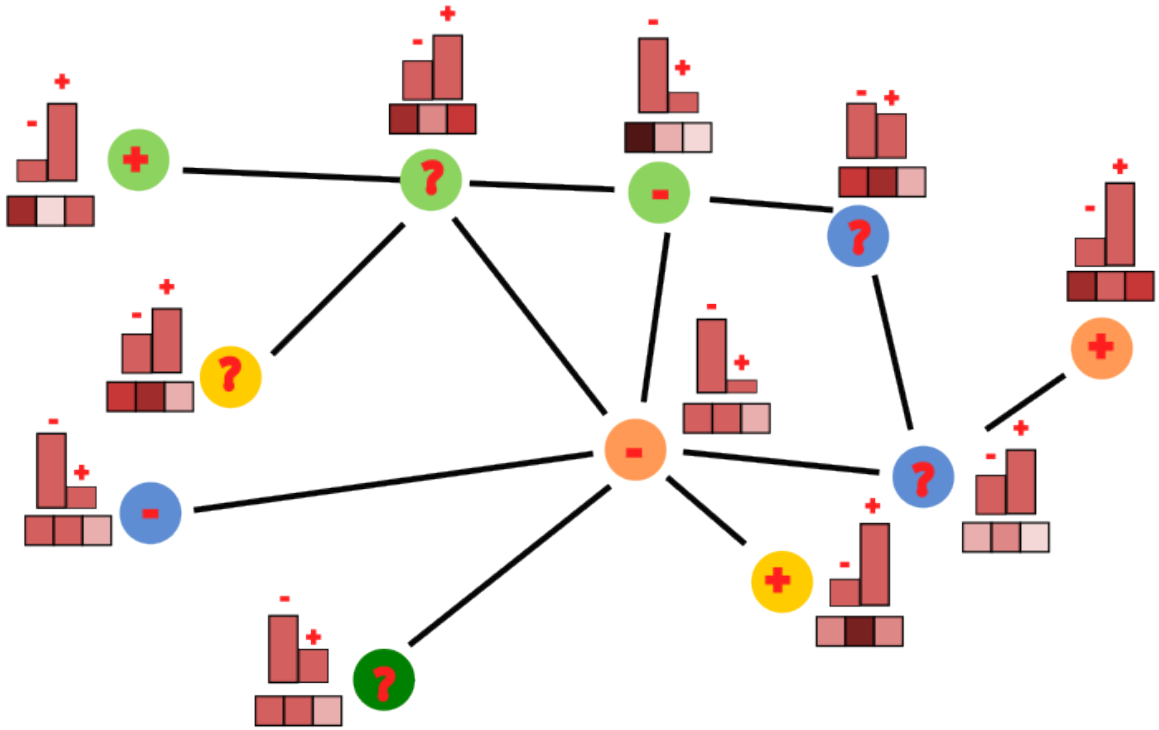
- Given: input graph with labeled (+,-) and unlabeled (?) nodes



- Given: input graph with labeled (+,-) and unlabeled (?) nodes
- GNN: compute node *embedding* (a.k.a. *representation*) vectors in $\mathbb{R}^d$



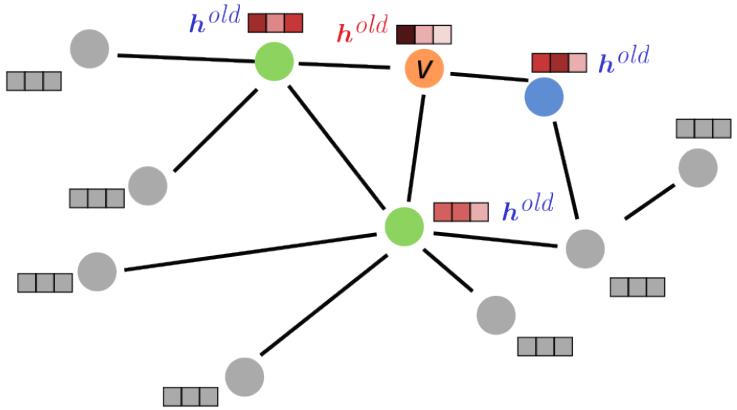
- Given: input graph with labeled (+,-) and unlabeled (?) nodes
- GNN: compute node *embedding* (a.k.a. *representation*) vectors in $\mathbb{R}^d$
- GNN: make probabilistic class predictions based on embeddings



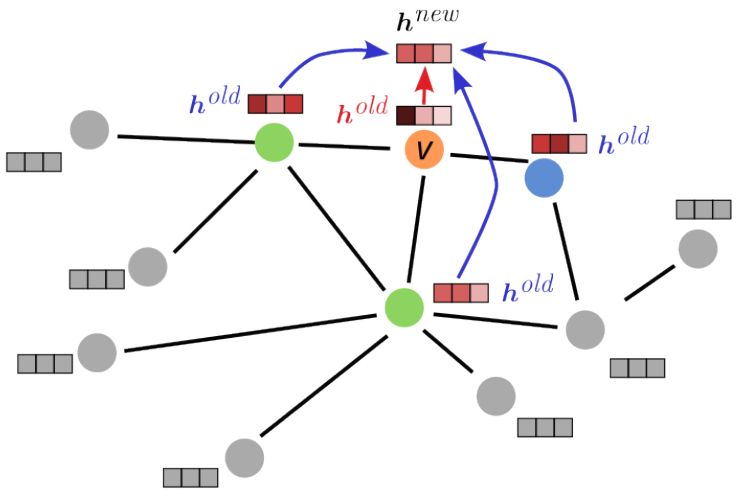
- Given: input graph with labeled (+,-) and unlabeled (?) nodes
- GNN: compute node *embedding* (a.k.a. *representation*) vectors in $\mathbb{R}^d$
- GNN: make probabilistic class predictions based on embeddings

➥Node embeddings also core element for graph classification and link prediction

GNNs compute node embeddings by *message passing layers* that iteratively update node embedding vectors $\boldsymbol{h}^{old}$ to $\boldsymbol{h}^{new}$:

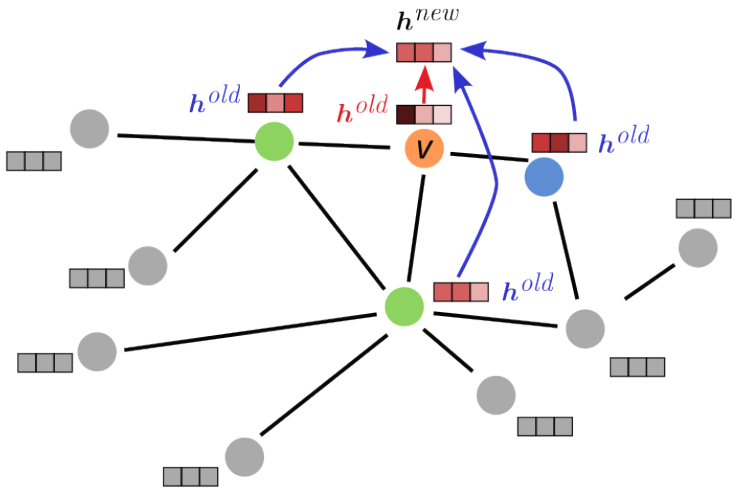


GNNs compute node embeddings by *message passing layers* that iteratively update node embedding vectors $\boldsymbol{h}^{old}$ to $\boldsymbol{h}^{new}$:



$$\boldsymbol{h}^{new}(v) := F(\boldsymbol{h}^{old}(v), \{\!\{\boldsymbol{h}^{old}(w) | w \in N(v)\}\!\}).$$

GNNs compute node embeddings by *message passing layers* that iteratively update node embedding vectors $\boldsymbol{h}^{old}$ to $\boldsymbol{h}^{new}$:



$$\boldsymbol{h}^{new}(v) := F(\boldsymbol{h}^{old}(v), \{\!\{\boldsymbol{h}^{old}(w) | w \in N(v)\}\!\}).$$

Key ingredients in F:

- *Aggregation* function to handle multiset $\{\!\{\boldsymbol{h}^{old}(w) | w \in N(v)\}\!\}$
- *Activation functions* for elementwise non-linear transformations

GNN-Architecture $\simeq$ design of function class $F()$. For example: GCN, GraphSage, GIN, GAT, ...

Expressivity

Expressivity: informal

Expressivity of a GNN architecture $\mathcal{M}$:

Existence of models $M \in \mathcal{M}$ that can distinguish nodes (or graphs) of different classes

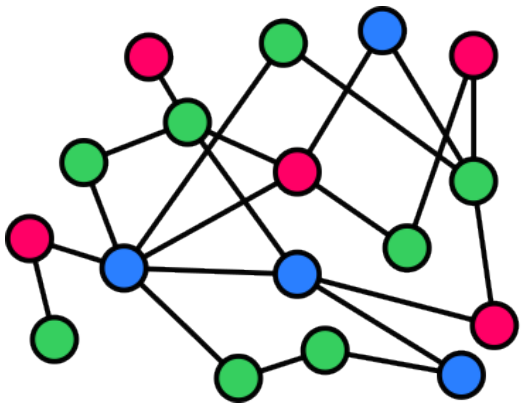
➥One GNN class is more expressive than another, if it can distinguish more classes

A formal logical formula:

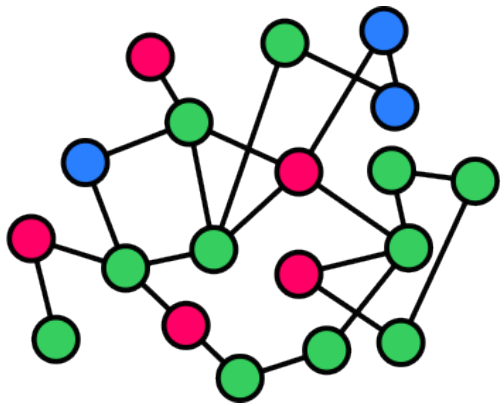
$$\phi := \exists^{\geq 2} v \, \exists^{\geq 1} w : \; red(v) \wedge blue(w) \wedge edge(v, w)$$

In natural languge: *there exist at least two red nodes that have at least one blue neighbor*

Two graphs that are distinguished by the property ϕ:



$\models \phi$



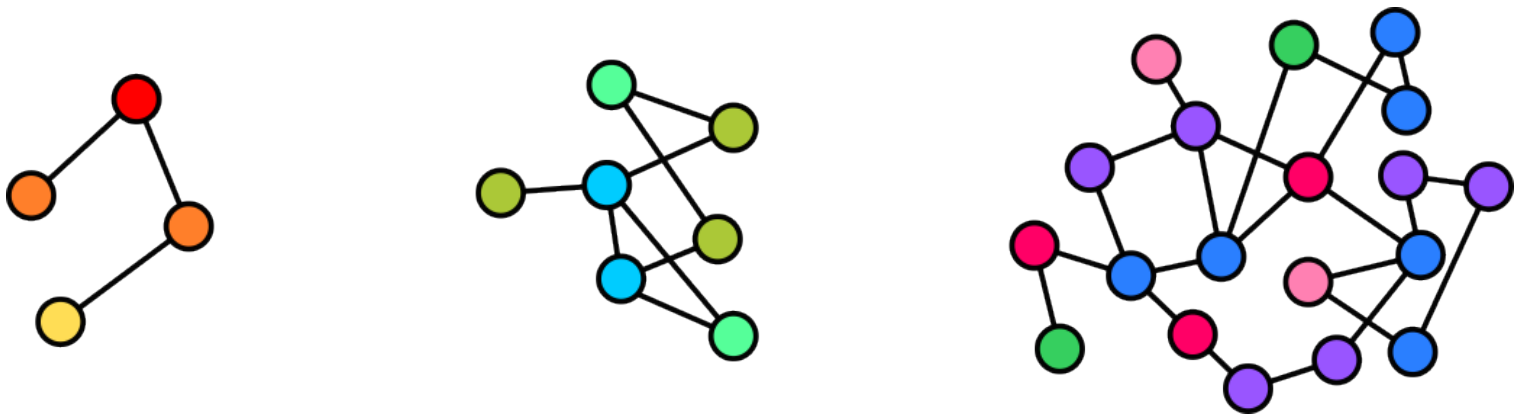
$\not\models \phi$

An Important Theorem

The class ACR-GNN can distinguish graphs according to properties expressible in the two-variable fragment of first-order logic with counting quantifiers

Barceló, Pablo, et al. "The logical expressiveness of graph neural networks." International conference on learning representations. 2020.

A given model M partitions the nodes of any graph into equivalence classes of nodes with identical embeddings:

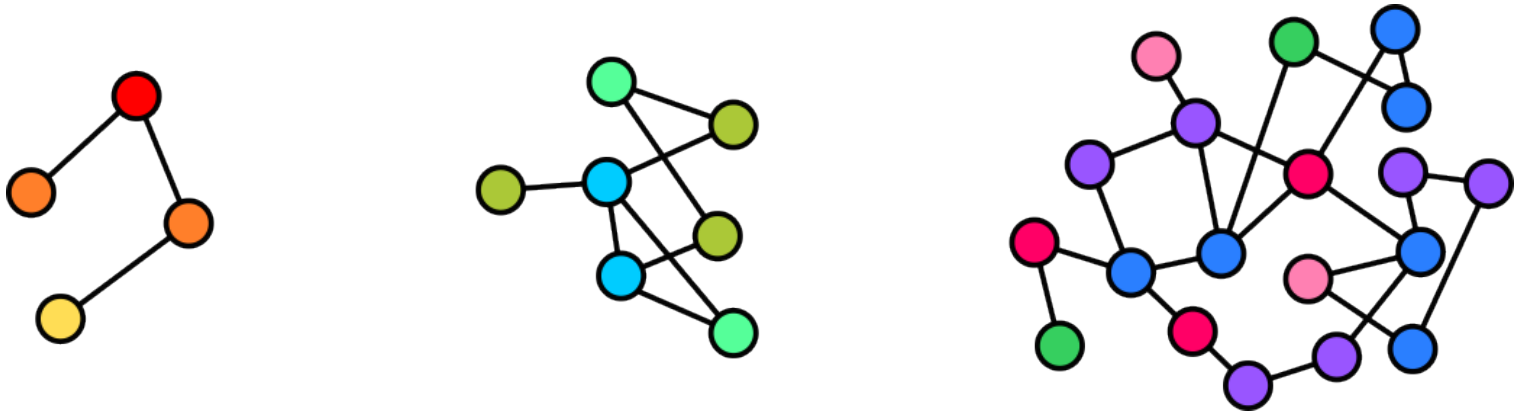


This leads to *"at least as expressive as" relationships*:

Uniform version: model class $\mathcal{M}$ is at least as expressive as $\mathcal{M}'$, if for all $M' \in \mathcal{M}'$ there exists $M \in \mathcal{M}$ such that for all graphs G: the partition defined by M on G refines the partition defined by M' on G.

Local version: model class $\mathcal{M}$ is at least as expressive as $\mathcal{M}'$, if for all $M' \in \mathcal{M}'$ and graphs G there exists $M \in \mathcal{M}$ such that the partition defined by M on G refines the partition defined by M' on G.

A given model M partitions the nodes of any graph into equivalence classes of nodes with identical embeddings:



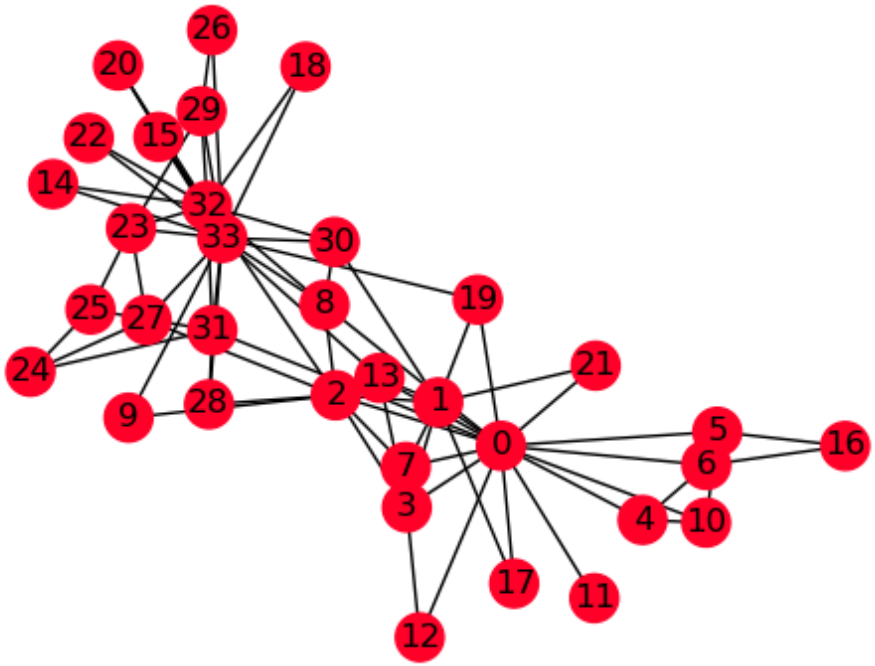
This leads to *"at least as expressive as" relationships*:

Uniform version: model class $\mathcal{M}$ is at least as expressive as $\mathcal{M}'$, if for all $M' \in \mathcal{M}'$ there exists $M \in \mathcal{M}$ such that for all graphs G: the partition defined by M on G refines the partition defined by M' on G.

Local version: model class $\mathcal{M}$ is at least as expressive as $\mathcal{M}'$, if for all $M' \in \mathcal{M}'$ and graphs G there exists $M \in \mathcal{M}$ such that the partition defined by M on G refines the partition defined by M' on G.

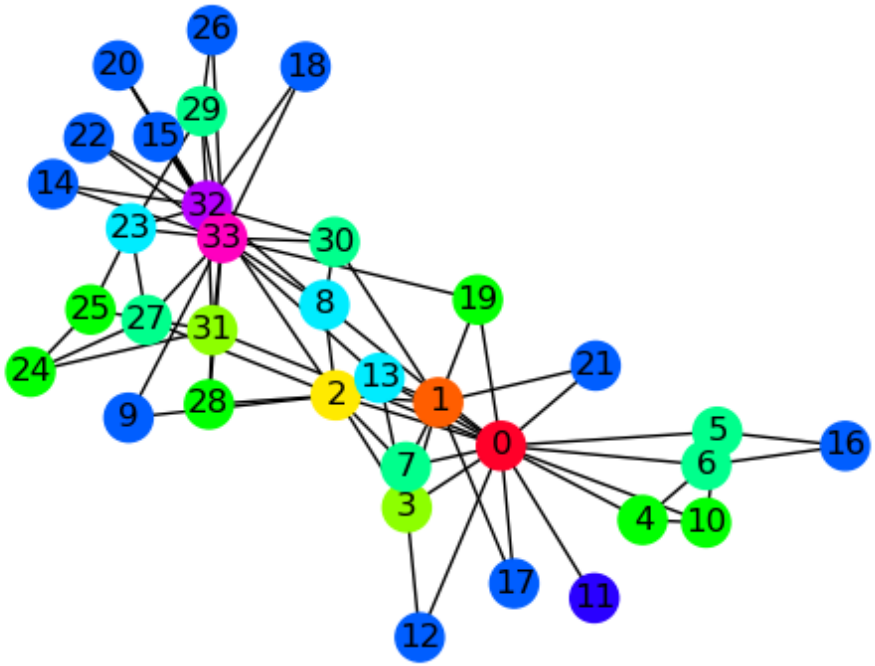
➥This makes mostly sense when initial node features are only categorical (represented as one out of finitely many possible "initial colors"). Numerical features: with sufficient precision, already the initial features are unique, defining a maximally fine partition.

Powerful benchmark for node partitioning: Weisfeiler-Lehman color refinement



Initialize colors: encode node properties

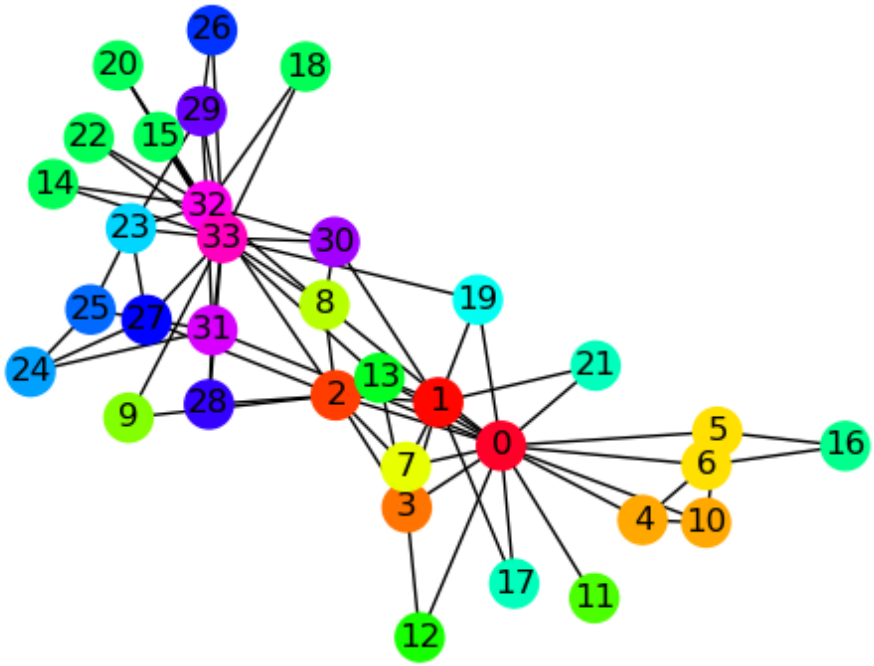
Powerful benchmark for node partitioning: Weisfeiler-Lehman color refinement



Initialize colors: encode node properties

Refine: assign unique color as a function of node's own color, and the multiset of colors of its neighbors

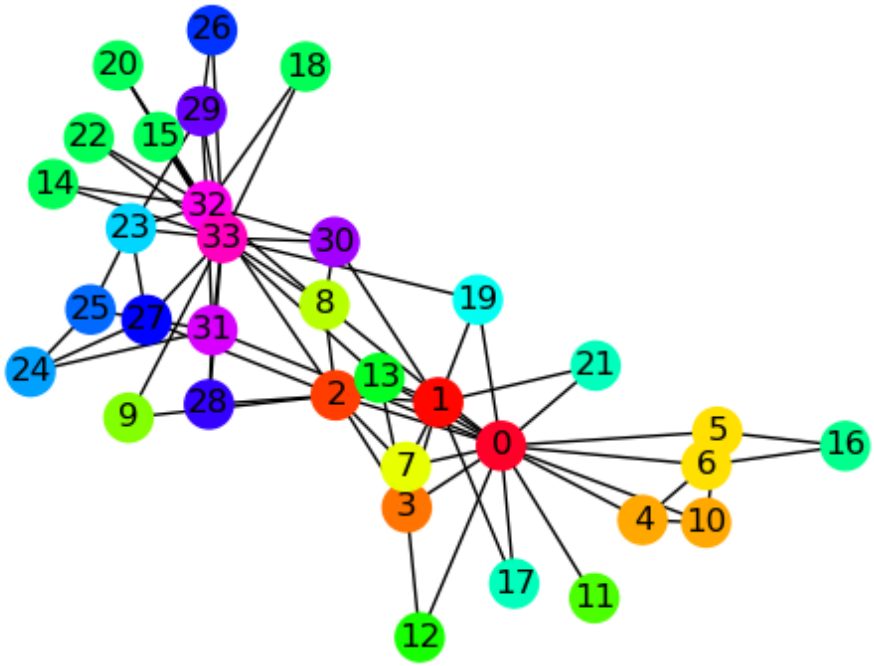
Powerful benchmark for node partitioning: Weisfeiler-Lehman color refinement



Initialize colors: encode node properties

Refine: assign unique color as a function of node's own color, and the multiset of colors of its neighbors

Powerful benchmark for node partitioning: Weisfeiler-Lehman color refinement

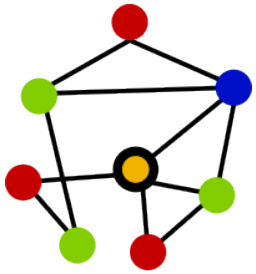


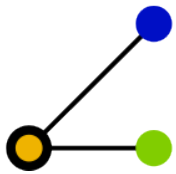
Initialize colors: encode node properties

Refine: assign unique color as a function of node's own color, and the multiset of colors of its neighbors

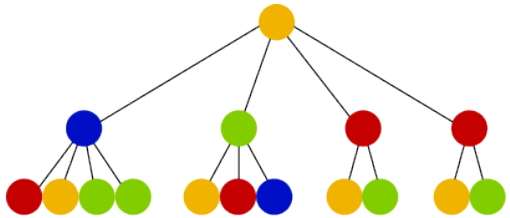
Terminate: no further refinement of color-induced node partition

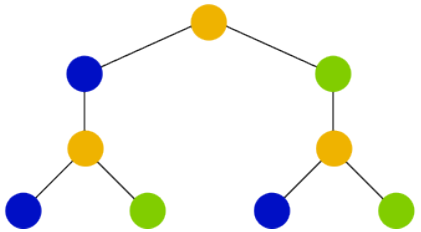
Ego graphs: graphs with a designated *root* node:





Tree unfoldings of depth *l*:





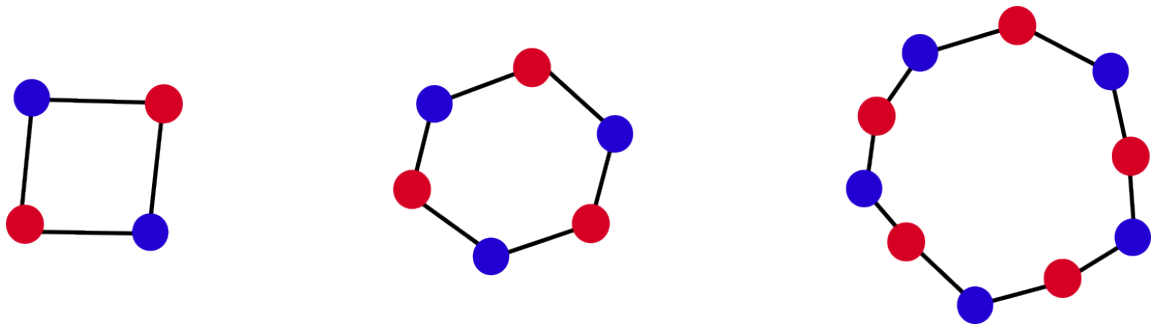
Tree unfolding of depth *l* of an ego graph:

- nodes correspond to paths of length $\leq l$ starting from the root node
- the children of a path (node) are all the one-step extensions of the path
- a path (node) is labeled with the feature vector ("color") of the last node on the path

When two nodes in a graph (or different graphs) have identical depth-l tree unfoldings:

- their colors after l rounds of WL refinement are the same
- their embedding vectors computed by l layer GNNs are the same

Example



All blue (red) nodes in these graphs have identical unfolding trees for any depth.
➥WL colors or GNN embeddings can never refine the initial partition into red and blue nodes (or the red nodes in one graph from the red nodes in another graph)

Note: ACR-GNNs can do that: red node in one graph distinguished from red node in another graph by formula

$$red(v) \wedge \exists^{\geq 3} w\, blue(w)$$

How so? Answer: the *readout* element in ACR-GNNs goes beyond the local message passing updates, and hence beyond the dependence on only the depth-l tree unfoldings.

GNN expressivity upper bound

For WL also the converse holds:

- when two nodes have different depth-l tree unfoldings, then they receive a different color in the lth round of WL refinement.

This is because, by definition, the WL function

$$(color^{old}(v), \{\!\{color^{old}(w) | w \in N(v)\}\!\}) \mapsto color^{new}(v)$$

is **injective**.

In GNNs:

$$(\boldsymbol{h}^{old}(v), \{\!\{\boldsymbol{h}^{old}(w) | w \in N(v)\}\!\}) \mapsto \boldsymbol{h}^{new}(v)$$

via a *continuous*, *differentiable* function $F()$.

➥Can this $F()$ also be injective?

Example



	$\boldsymbol{h}^{old}$
●	$(1,0,0)$
●	$(0,1,0)$
●	$(0,0,1)$

WL: the two yellow nodes receive different colors in the next update.

GNN:

If $F(\boldsymbol{h}^{old}(v), \{\!\{\boldsymbol{h}^{old}(w) | w \in N(v)\}\!\})$ applies

- *mean aggregation* to $\{\!\{\boldsymbol{h}^{old}(w) | w \in N(v)\}\!\}$: $\boldsymbol{h}^{new}$ will be the same for the two different yellow nodes.
- *sum aggregation*: $\boldsymbol{h}^{new}$ can be different for the two different yellow nodes.

➥A necessary requirement for expressivity is injectivity: the ability to map different multisets to different aggregates.

A Classic Theorem

There does not exist a continuous injective function from $\mathbb{R}^n$ to $\mathbb{R}^m$ if $n > m$.

Luitzen EJ Brouwer. Beweis der Invarianz des n-dimensionalen Gebiets. Mathematische Annalen, 71(3):305-313, 1911.

J. van Mill. Domain invariance. Encyclopedia of Mathematics. URL: http://encyclopediaofmath.org/index.php?title=Domain_invariance&oldid=16623.

This means that

$$(\boldsymbol{h}^{old}(v), \{\!\{\boldsymbol{h}^{old}(w) | w \in N(v)\}\!\}) \mapsto \boldsymbol{h}^{new}(v)$$

cannot be *continuous* and *injective* when

- the dimension of $\boldsymbol{h}^{new}(v)$ is fixed
- the multisets $\{\!\{\boldsymbol{h}^{old}(w) | w \in N(v)\}\!\}$ can be arbitrarily large

Graph Isomorphism Networks (GINs) are defined by layers of the form:

$$\boldsymbol{h}^{new}(v) := \text{MLP}((1+\epsilon)\boldsymbol{h}^{old}(v) + \text{SUM}\{\!\{\boldsymbol{h}^{old}(w) | w \in N(v)\}\!\})$$

- Sum aggregation!
- MLP: sequence of linear layers + activation functions

Folklore: Graph Isomorphism Networks (GINs) " generalize the Weisfeiler-Lehman test and hence achieve maximum discriminative power among GNNs " [Xu et al., 2019]

[Xu, Keyulu, et al. "How Powerful are Graph Neural Networks?." ICLR 2019]
[Sato, Ryoma. "A survey on the expressive power of graph neural networks." arXiv:2003.04078 (2020)]

Define: $\mathcal{T}^{(k)}$: set of all depth-k unfolding trees (for a fixed finite set of initial node colors).

Maximal *Bounded* Expressivity

For any **finite** set $\{T_1, \ldots, T_N\} \subset \mathcal{T}^{(k)}$: there exists a k-layer $M \in$ GIN with N-dimensional output and only using *ramp* activation functions s.t. for all $T \in \mathcal{T}^{(k)}$:

$$M(T)[i] = \begin{cases} 1 & \text{if } T = T_i \\ 0 & \text{otherwise} \end{cases}$$

➥ Essence of existing "GIN is maximally expressive" results.

➥ Can be shown by elementary, constructive proof by induction over k.

➥ This does *not imply* the *unbounded* expressivity result for ACR-GNN: the sets of graphs (nodes) that satisfy ϕ, and the set of graphs (nodes) that do not satisfy ϕ are usually both infinite.

Limits

What happens when graphs become bigger and bigger?



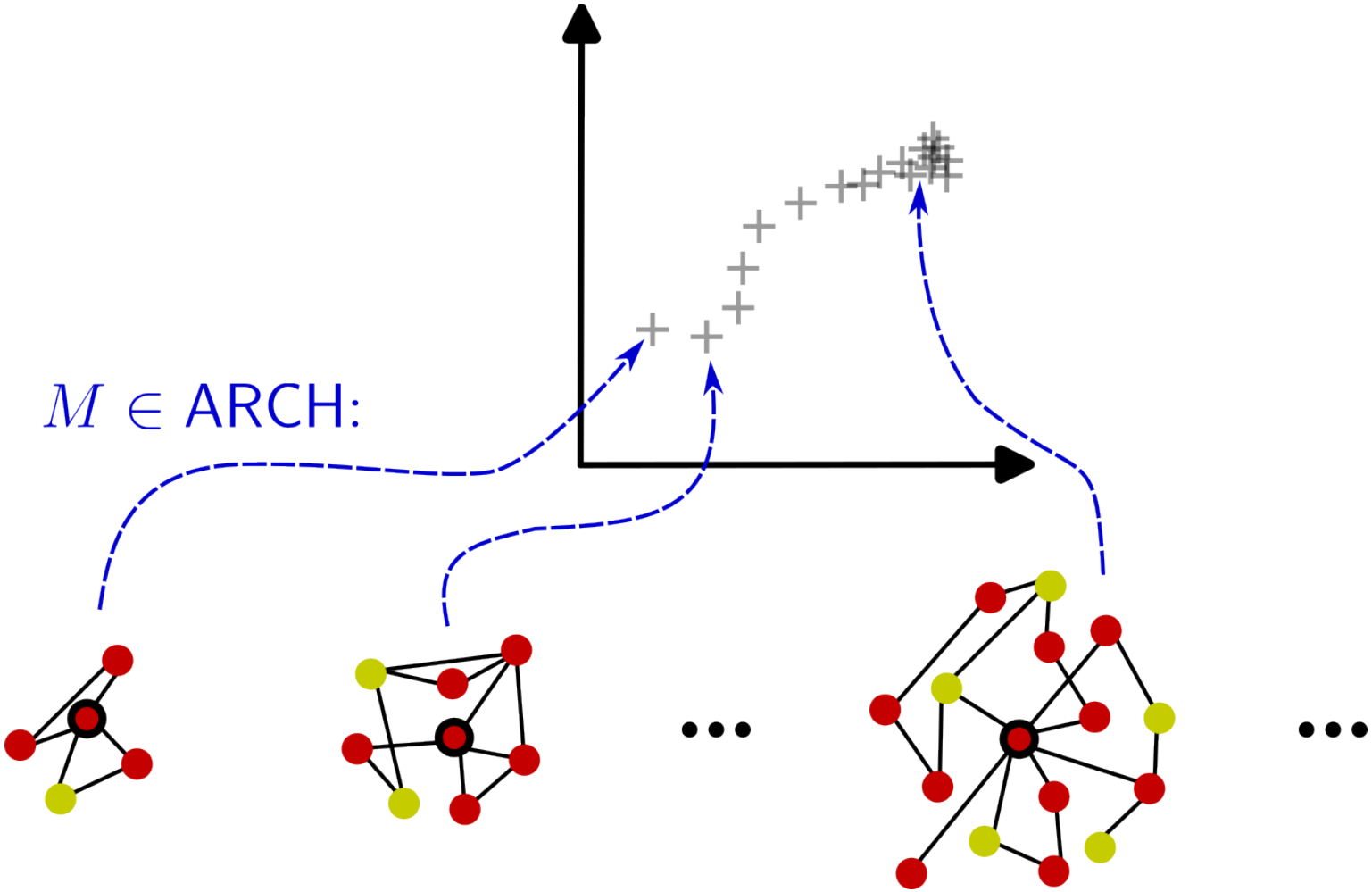
Do predictions converge?

Implications for

- Domain size generalization
- Robustness w.r.t. graph uncertainty
- ...

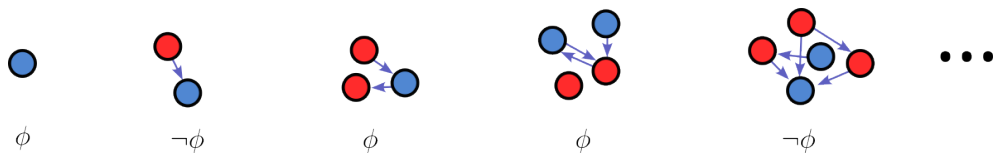
➥Rather limited existing theory, mostly based on the *graphon* model.

Embeddings of a designated **root node** computed by a GNN M from a class ARCH:

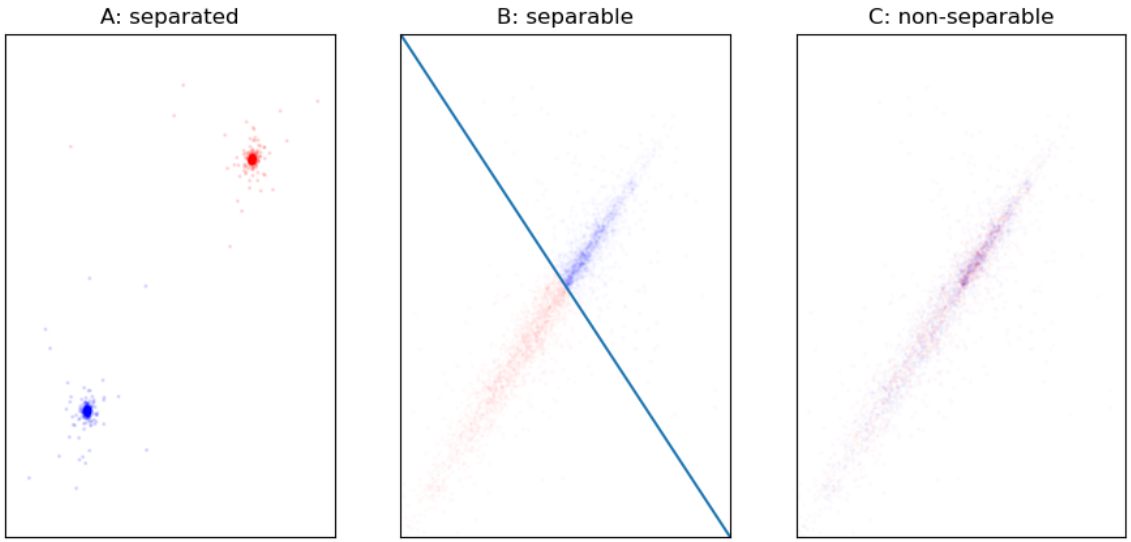


Question: under which conditions on ARCH and the input graph sequence are the embeddings guaranteed to converge?

Consider infinite sequence of *labeled* graphs:

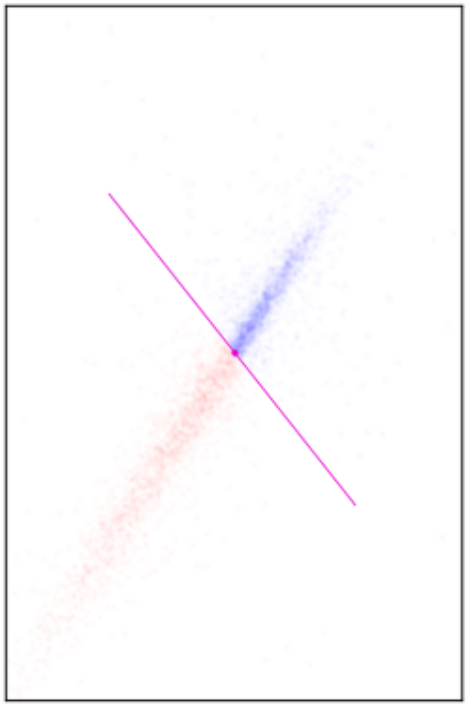


Types of embedding sequences ($\phi \sim$ red; $\neg\phi \sim$ blue):



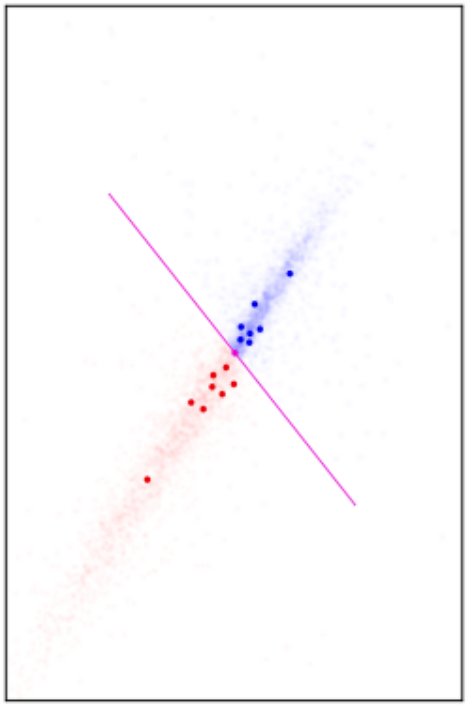
- A not convergent, classes separated
- B convergent, classes separable by discontinuous classification function (linear + *sign*, softmax + *argmax*,...)
- C convergent, classes not linearly separable

Convergence scenario B not good enough for robust size generalization:



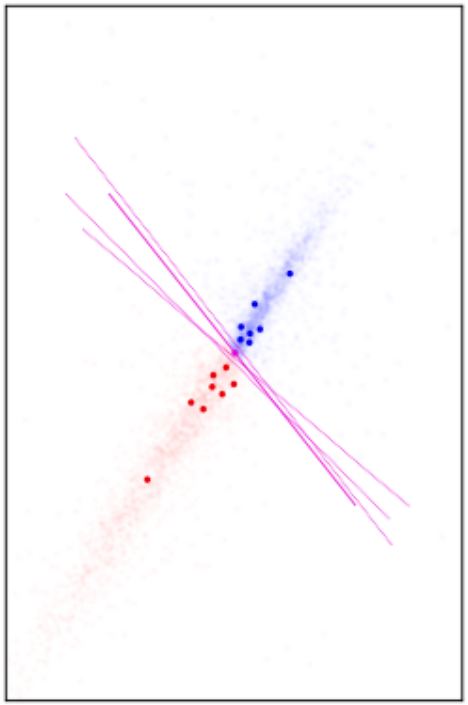
- Separating line must pass through unique limit point of sequence

Convergence scenario B not good enough for robust size generalization:



- Separating line must pass through unique limit point of sequence
- Finite training set

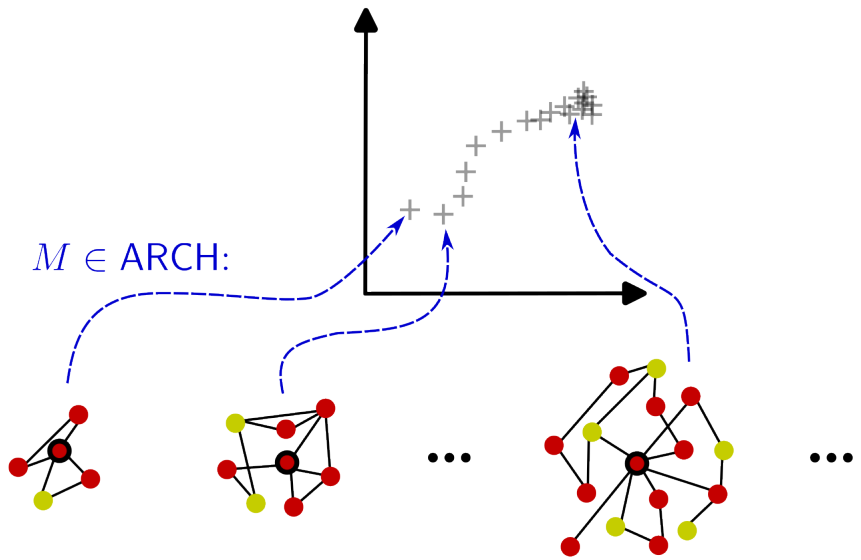
Convergence scenario B not good enough for robust size generalization:



- Separating line must pass through unique limit point of sequence
- Finite training set
- Decision boundary learned will almost surely not hit the limit point

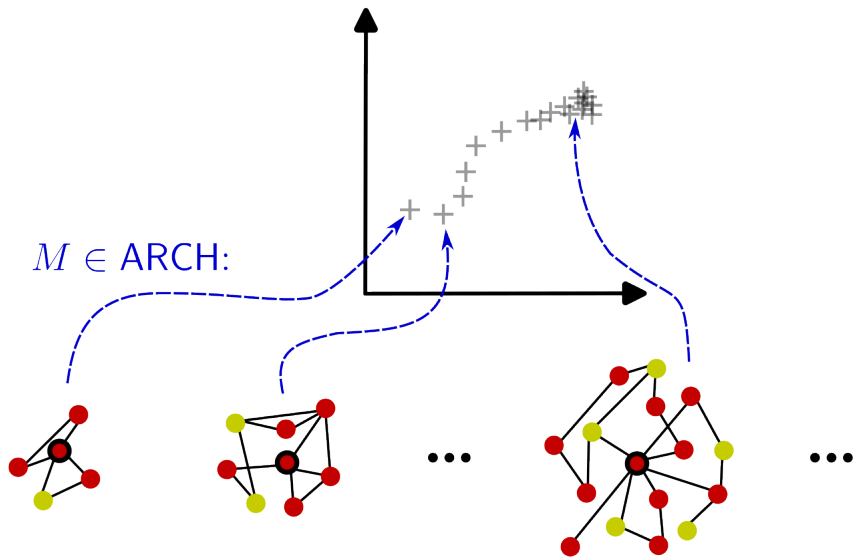
Remember our **Question:** under which conditions on ARCH and the input graph sequence are the embeddings guaranteed to converge?

➥Positive answers for this question are negative results for separation-expressivity of ARCH!



Remember our **Question:** under which conditions on ARCH and the input graph sequence are the embeddings guaranteed to converge?

➥Positive answers for this question are negative results for separation-expressivity of ARCH!

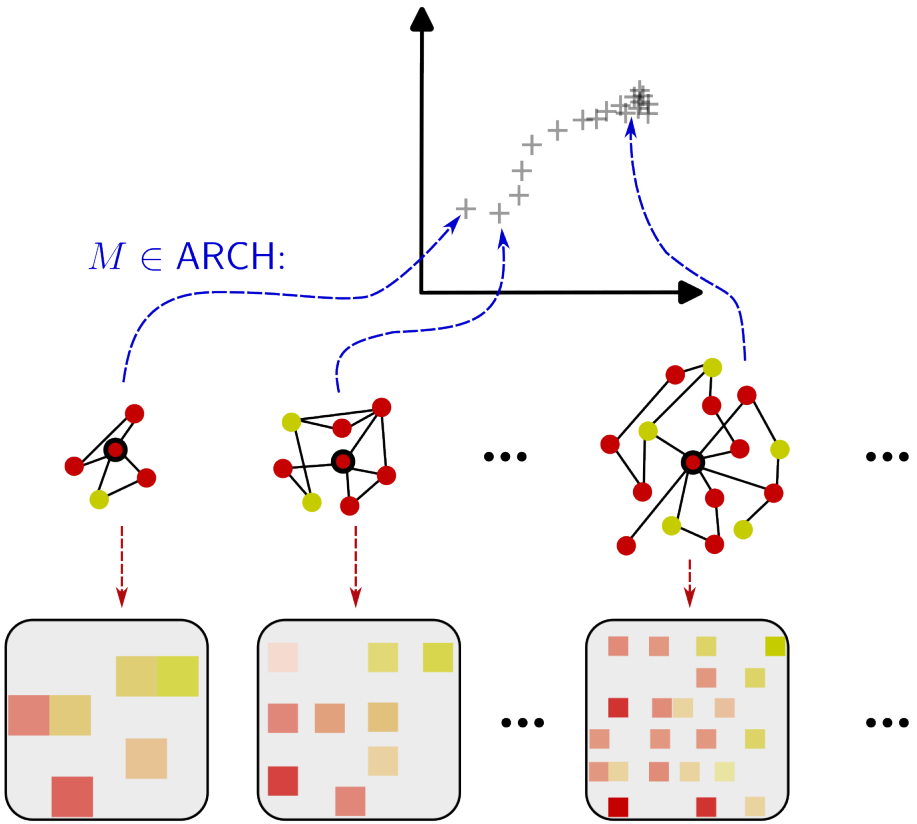


Answer type 1:

For “ad-hoc” defined sequences: do embeddings defined by $M \in$ ARCH converge?

Remember our **Question:** under which conditions on ARCH and the input graph sequence are the embeddings guaranteed to converge?

➥Positive answers for this question are negative results for separation-expressivity of ARCH!



Answer type 1:

For "ad-hoc" defined sequences: do embeddings defined by $M \in$ ARCH converge?

Answer type 2:

- Map (rooted) graphs into a metric space with metric d
- Condition on the input graph sequence: convergent (or: Cauchy) w.r.t. d
- Then: positive answer to our question: embeddings defined by $M \in$ ARCH are **continuous** w.r.t. d

The class LMEAN

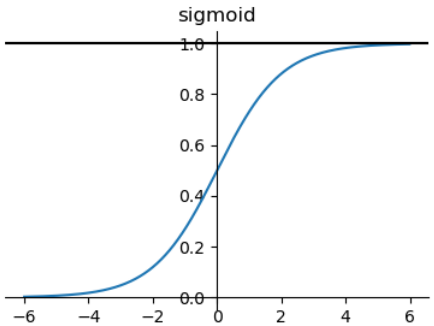
Any function

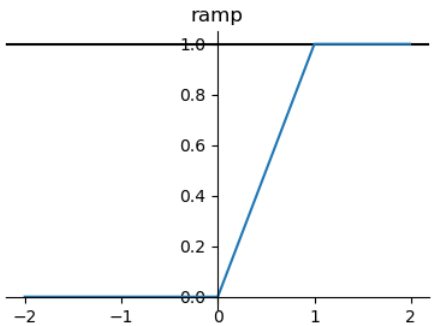
$$\alpha : \mathbb{R} \to \mathbb{R}$$

Special properties:

- *Monotonicity*
- *Lipschitz continuity*
- *Squashing*: monotone, and exist $l, u \in \mathbb{R}$: $\lim_{x \to -\infty} \alpha(x) = l$ and $\lim_{x \to \infty} \alpha(x) = u$.
- *Fast Squashing*: squashing, and for all $r > 0$:

$$\alpha(nr) = u - o(n^{-\infty}) \text{ and } \alpha(-nr) = l + o(n^{-\infty})$$





The *sigmoid* and *ramp* activation functions are Lipschitz and fast squashing.

A generic class for *mean*-aggregation:

$$\boldsymbol{h}^{new}(v) := F(\boldsymbol{h}^{old}(v), \text{MEAN}\{\!\{\boldsymbol{h}^{old}(w) | w \in N(v)\}\!\}).$$

where F is Lipschitz continuous.

- F can contain:
  - concatenation or summation of its two arguments
  - linear transformations
  - activation functions: *sigmoid*,*ramp*,*relu*,...
- Important example: GraphSage

Goal: define a metric on unfolding trees $\mathcal{T}^{(k)}$, such that all $M \in$ LMEAN are continuous w.r.t. this metric

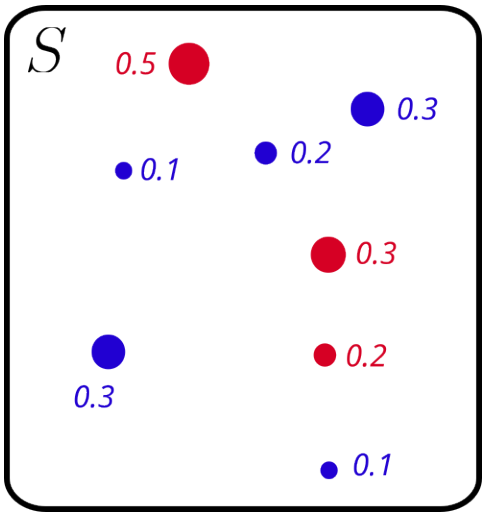
More precisely: we will construct a *pseudo-metric*, i.e. can have for trees $T \neq T'$: $d(T, T') = 0$.

General construction:

$$\boxed{\text{Metric space } (S, d)} \mapsto \boxed{\text{Metric on space of probability distributions over } S}$$

General construction:

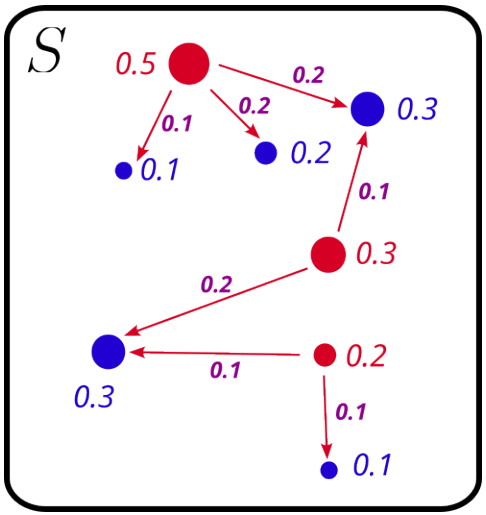
Metric space (S, d) $\mapsto$ Metric on space of probability distributions over S



- Two probability distributions P, Q on S (here discrete: finitely many probability masses on points of S).

General construction:

Metric space (S, d) $\mapsto$ Metric on space of probability distributions over S

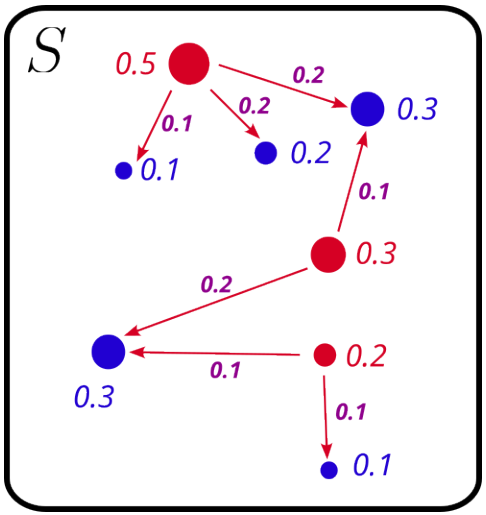


- Two probability distributions P, Q on S (here discrete: finitely many probability masses on points of S).
- Define $d_W(P, Q)$ as the minimal cost of transformations of P into Q. Cost of transformation:

$$\sum \text{probability mass} \times \text{distance}$$

General construction:

Metric space (S, d) $\mapsto$ Metric on space of probability distributions over S



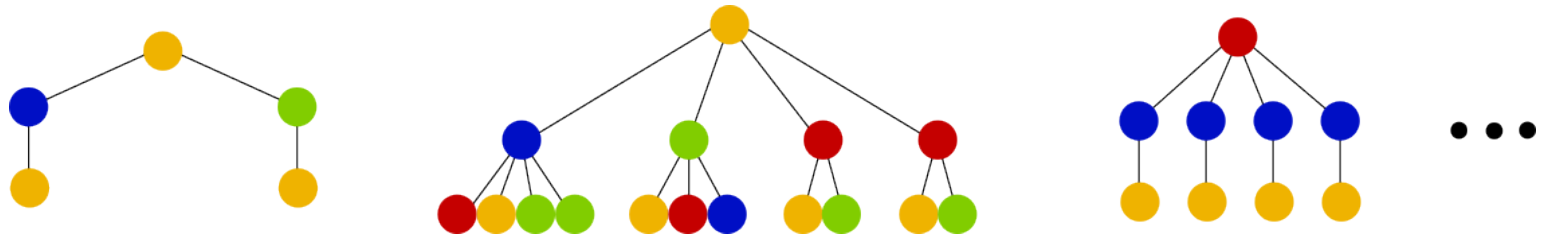
- Two probability distributions P, Q on S (here discrete: finitely many probability masses on points of S).
- Define $d_W(P, Q)$ as the minimal cost of transformations of P into Q. Cost of transformation:

$$\sum \text{probability mass} \times \text{distance}$$

- This is symmetric in P and Q
- Formally expressed as solution to a linear programming problem
- Also known as Earth Mover distance

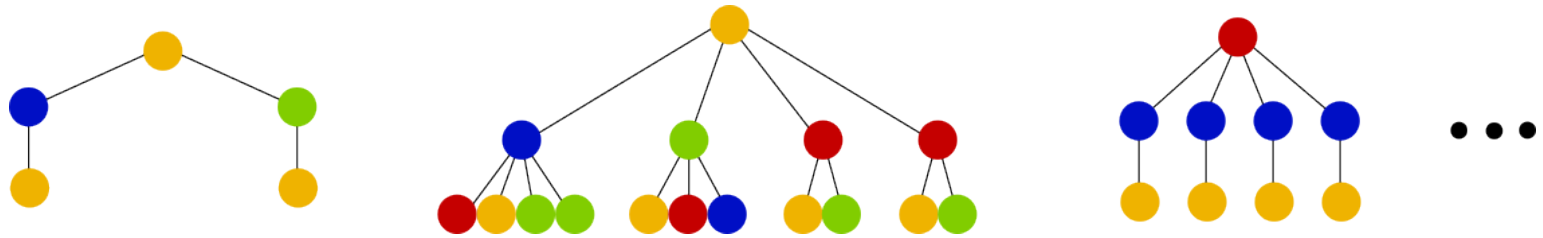
$\mathcal{T}^{(k)}$: set of all (unfolding) trees of depth k (fixed categorical node attribute)

$\mathcal{T}^{(2)}$:



$\mathcal{T}^{(k)}$: set of all (unfolding) trees of depth k (fixed categorical node attribute)

$\mathcal{T}^{(2)}$:



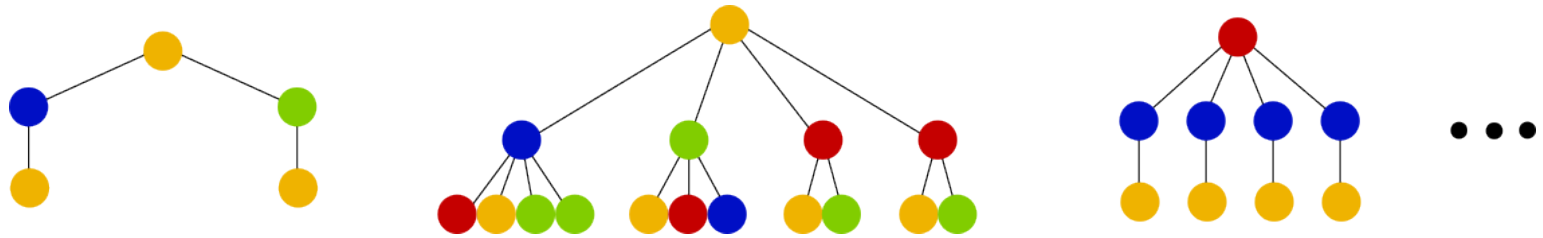
Define by induction pseudo-metric $d_W^{(k)}$ on $\mathcal{T}^{(k)}$:

$k = 0$:

- $\mathcal{T}^{(0)}$ is the finite set of isolated nodes
- Define $d_W^{(0)}$ as the 0-1 metric

$\mathcal{T}^{(k)}$: set of all (unfolding) trees of depth k (fixed categorical node attribute)

$\mathcal{T}^{(2)}$:



Define by induction pseudo-metric $d_W^{(k)}$ on $\mathcal{T}^{(k)}$:

$k = 0$:

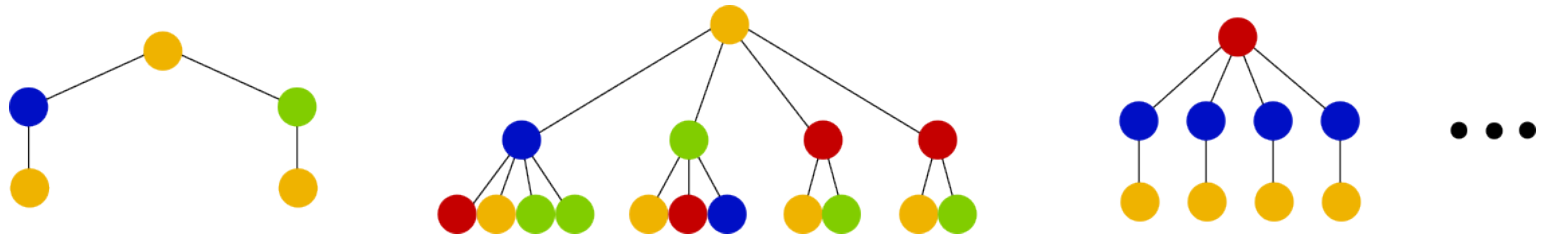
- $\mathcal{T}^{(0)}$ is the finite set of isolated nodes
- Define $d_W^{(0)}$ as the 0-1 metric

$k > 0$:



$\mathcal{T}^{(k)}$: set of all (unfolding) trees of depth k (fixed categorical node attribute)

$\mathcal{T}^{(2)}$:

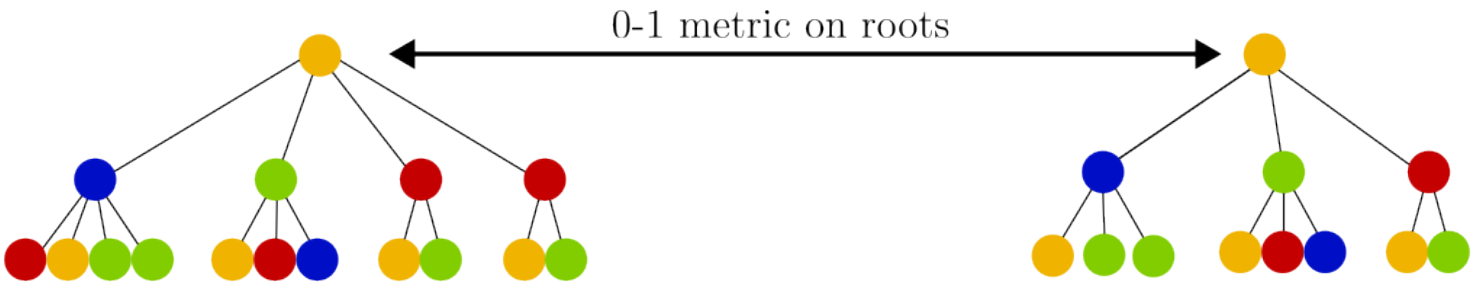


Define by induction pseudo-metric $d_W^{(k)}$ on $\mathcal{T}^{(k)}$:

$k = 0$:

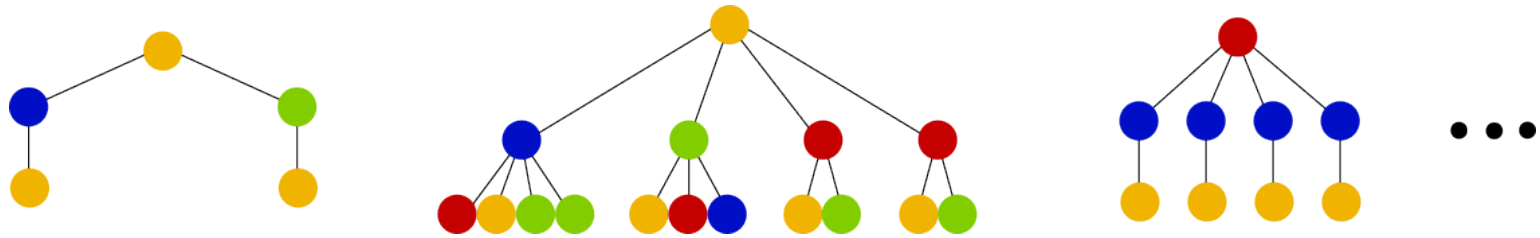
- $\mathcal{T}^{(0)}$ is the finite set of isolated nodes
- Define $d_W^{(0)}$ as the 0-1 metric

$k > 0$:



$\mathcal{T}^{(k)}$: set of all (unfolding) trees of depth k (fixed categorical node attribute)

$\mathcal{T}^{(2)}$:

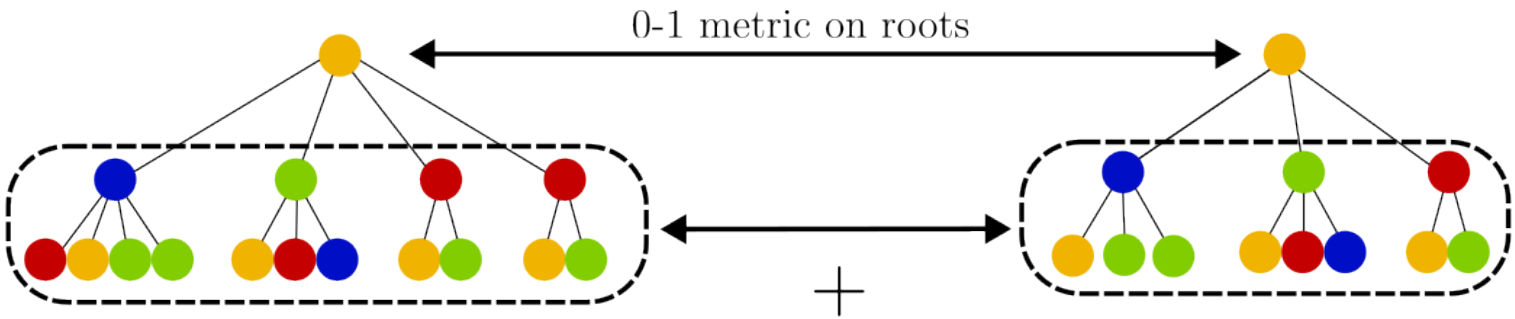


Define by induction pseudo-metric $d_W^{(k)}$ on $\mathcal{T}^{(k)}$:

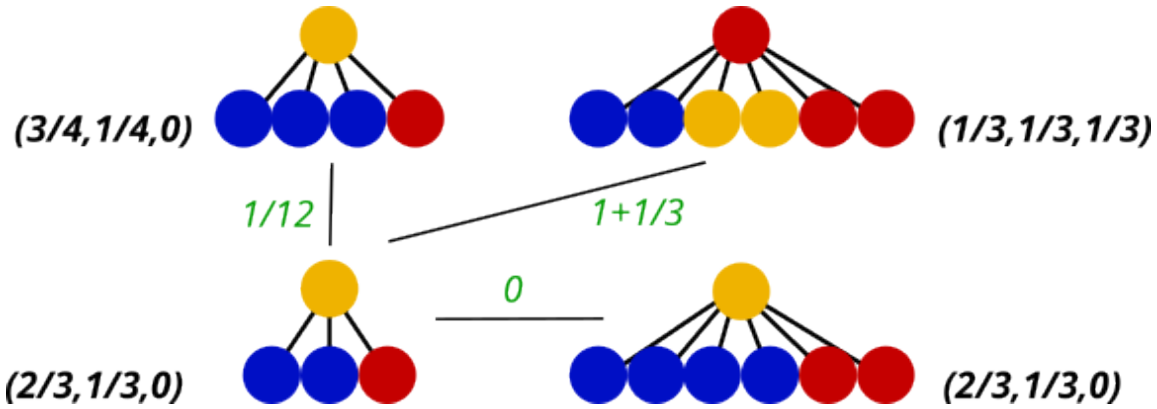
$k = 0$:

- $\mathcal{T}^{(0)}$ is the finite set of isolated nodes
- Define $d_W^{(0)}$ as the 0-1 metric

$k > 0$:



Wasserstein distance of the sub-tree distribution on $\mathcal{T}^{(k-1)}$ w.r.t. $d_W^{(k-1)}$



4 trees in $\mathcal{T}^{(1)}$ with probability distribution of child sets in $\mathcal{T}^{(0)}$:

$$(\mathbf{P}(blue), \mathbf{P}(red), \mathbf{P}(yellow))$$

and selected values of $d_W^{(1)}$

Theorem

For all $M \in \text{LMEAN}$, and all $k \geq 0$: the mapping

$$M : \mathcal{T}^{(k)} \to \mathbb{R}^d$$

is Lipschitz-continuous w.r.t. $d_W^{(k)}$.

Theorem

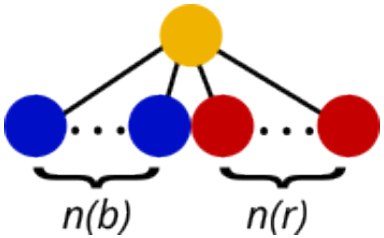
For all $M \in$ LMEAN, and all $k \geq 0$: the mapping

$$M : \mathcal{T}^{(k)} \to \mathbb{R}^d$$

is Lipschitz-continuous w.r.t. $d_W^{(k)}$.

What about GIN?

A sequence that is convergent according to $d_W^{(1)}$:

T_n: 

with $n(b), n(r) = \begin{cases} n, n & n \text{ even} \\ n, n+1 & n \text{ odd} \end{cases}$

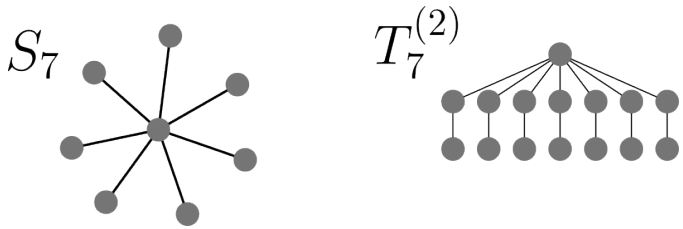
➥ there exists $M \in$ GIN with non-convergent embeddings: with *sum* can construct feature $n(r) - n(b)$ that oscillates between 1 and 0 as $n \to \infty$.

Open problem: is there a (natural, useful, ...) metric on $\mathcal{T}^{(k)}$ according to which all $M \in$ GIN are continuous?

Even/Odd Challenge

We consider:

- Plain graphs: no node attributes
- Define *even*/*odd* node class: even/odd number of neighbors
- Let S_n be the n-node *star* graph, and $T_n^{(k)}$ the depth-k unfolding tree of S_n's central node



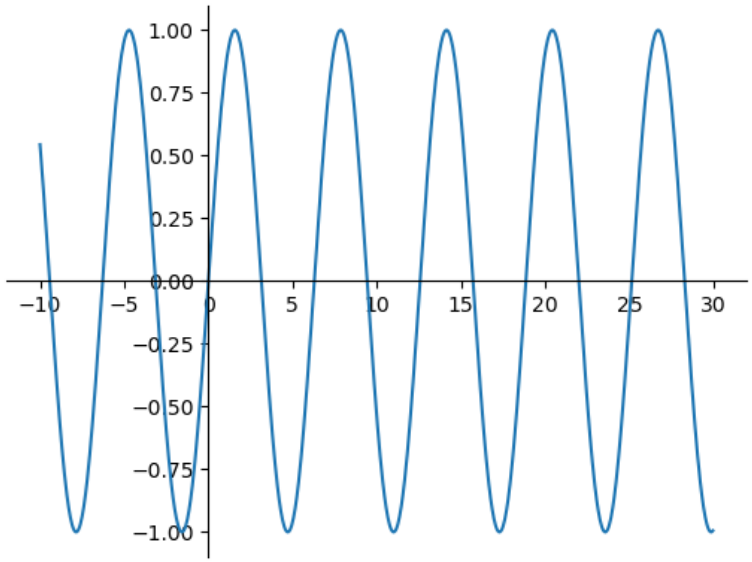
Theorem

Let $M \in$ GIN with k layers and only fast squashing activation functions. Then $M(T_n^{(k)})$ $(n = 1, 2, \ldots)$ is convergent.

Note: ReLU activation not covered, but unlikely to make a difference.

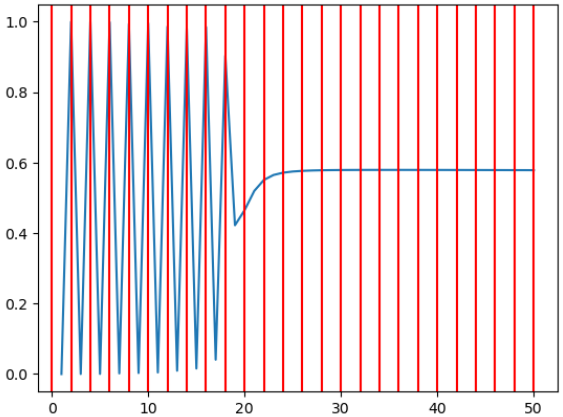
sine activation

Consider *sine* as an activation function:

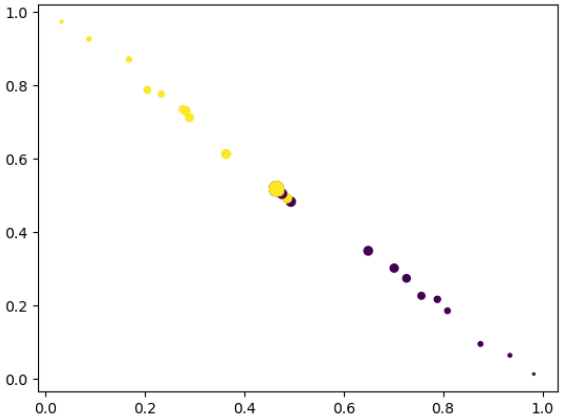


(or any other periodic function): there exists $M \in \text{GIN}$ with *sine* activation that separates the classes *even*/*odd*.

Training data: $S_1, \ldots, S_{30}$
Test data: $S_1, \ldots, S_{50}$
Prediction target: *even* class of center node

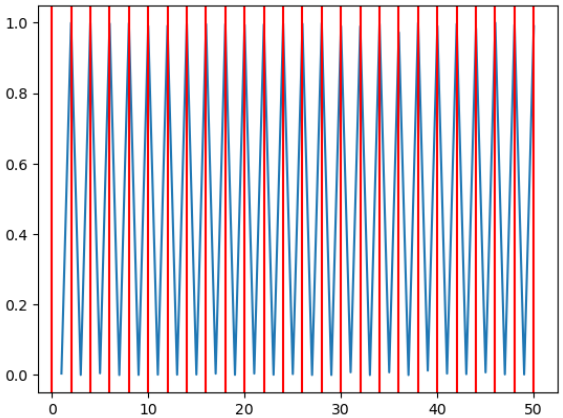


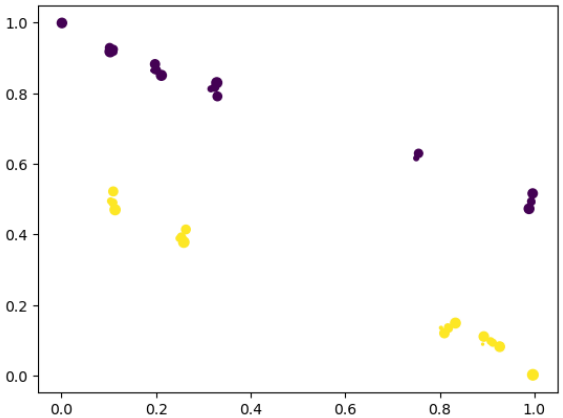
Predicted probability of *even* class



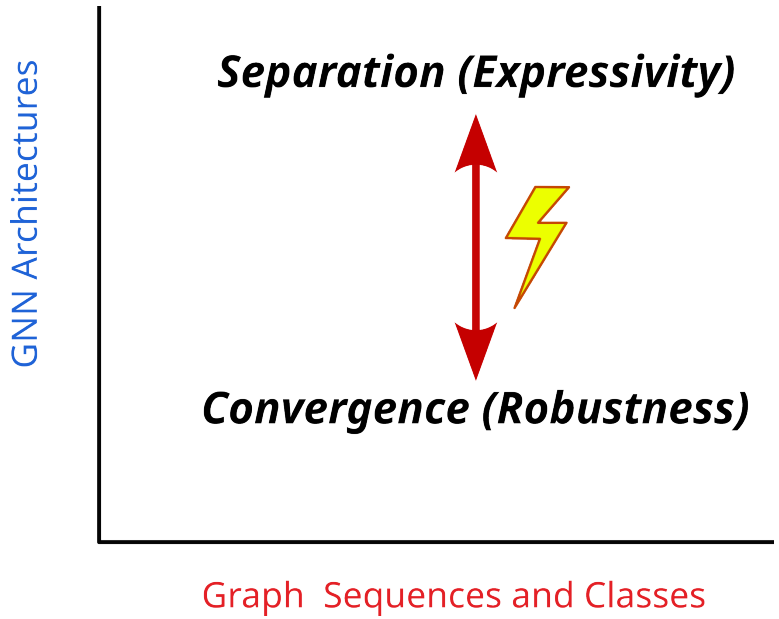
Embeddings at final 2-dimensional layer

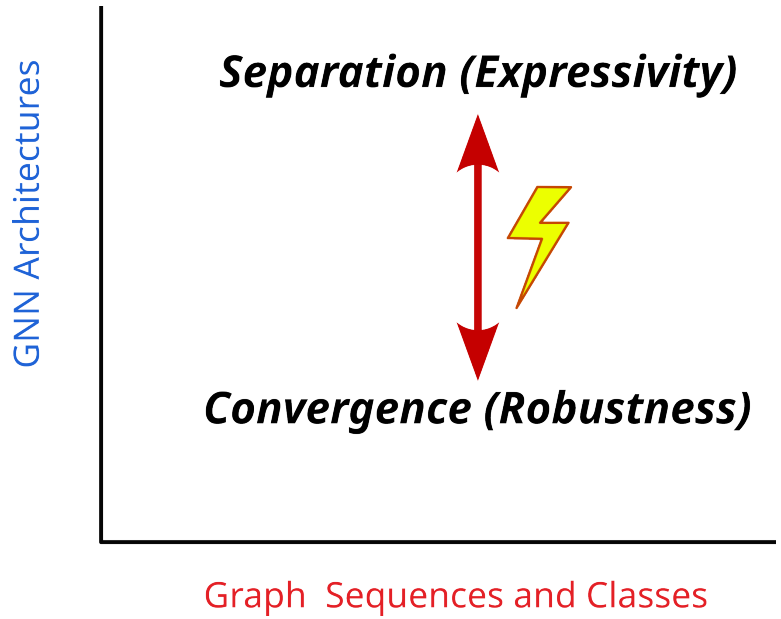
1-layer GIN trained with *sigmoid* activation (MLP: 3 hidden layers with dimensions 128,128,2)





1-layer GIN trained with *sine* activation (MLP: 2 hidden layers with dimensions 4,2)


GNN Architectures
Separation (Expressivity)
Convergence (Robustness)
Graph Sequences and Classes



- For expressivity: injectivity is not enough, need separation!
- Mean vs. Sum aggregation: robustness vs. expressivity
- GIN expressivity: “maximal” only in the bounded expressivity sense
- Functional properties of activation functions can play a big role: squashing vs. periodic, . . .