Learning and Reasoning with Graph Data: Integrating SRL and GNN

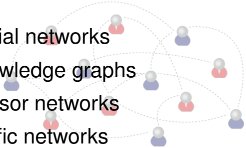
Manfred Jaeger

Aalborg University

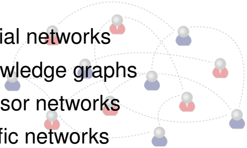
- Learning and reasoning with graphs: from logic to graph neural networks
- A few notes on GNNs
- A few notes on SRL
- Relational Bayesian Networks
- GNN-RBN integration

Real-world networks

- Social networks
- Knowledge graphs
- Sensor networks
- Traffic networks
- ...



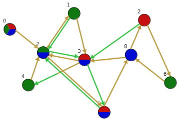
Real-world networks



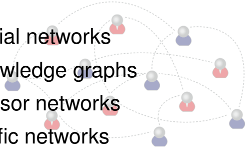
- Social networks
- Knowledge graphs
- Sensor networks
- Traffic networks
- ...

Model

Abstract: graphs



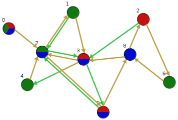
Real-world networks



- Social networks
- Knowledge graphs
- Sensor networks
- Traffic networks
- ...

Model

Abstract: graphs



Semantics

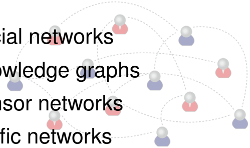
Predicate logic (relational)

$\forall x(r(x) \rightarrow \exists y(e(x,y) \wedge b(y)))$

$\exists z, x, y \neg(e(x,y) \wedge e(x,z) \wedge (e(y,z))$

...

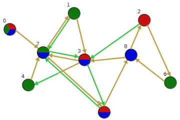
Real-world networks



- Social networks
- Knowledge graphs
- Sensor networks
- Traffic networks
- ...

Model

Abstract: graphs



Semantics

Predicate logic (relational)

$\forall x(r(x) \rightarrow \exists y(e(x,y) \wedge b(y)))$

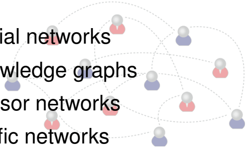
$\exists z, x, y \neg(e(x,y) \wedge e(x,z) \wedge (e(y,z))$

...

$+$*uncertainty*

- Probabilistic Logic
- Statistical (Random) Graph Theory
- Statistical Relational Learning (Probabilistic Logic Learning)
- Graph Learning and Mining
- Graph Neural Networks

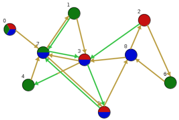
Real-world networks



- Social networks
- Knowledge graphs
- Sensor networks
- Traffic networks
- ...

Model ↓

Abstract: graphs



Semantics ↑

Predicate logic (relational)

$\forall x(r(x) \rightarrow \exists y(e(x,y) \wedge b(y)))$

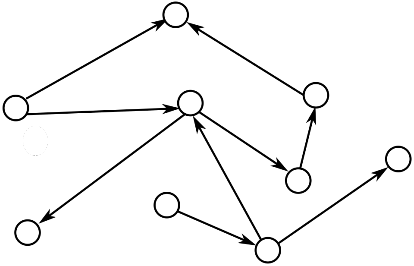
$\exists z, x, y \neg(e(x,y) \wedge e(x,z) \wedge (e(y,z))$

...

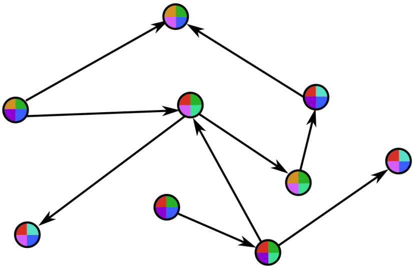
$+$*uncertainty* →

- Probabilistic Logic
- Statistical (Random) Graph Theory
- Statistical Relational Learning (Probabilistic Logic Learning)
- Graph Learning and Mining
- Graph Neural Networks

Graph Representations

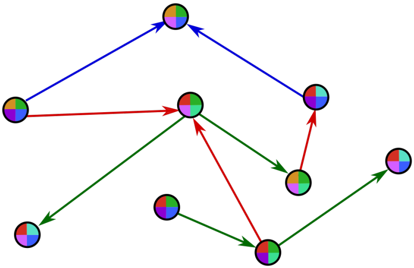


Graph: (V, E)



Graph: (V, E)

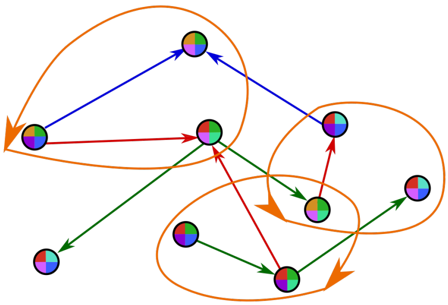
Attributed graph: $(V, E, \boldsymbol{A})$. Node attributes $\boldsymbol{A}$: *Boolean*, *categorical*, or *numeric*



Graph: (V, E)

Attributed graph: $(V, E, \boldsymbol{A})$. Node attributes $\boldsymbol{A}$: *Boolean*, *categorical*, or *numeric*

Attributed multirelational graph: $(V, \boldsymbol{E}, \boldsymbol{A})$. $\boldsymbol{E}$: set of different edge relations

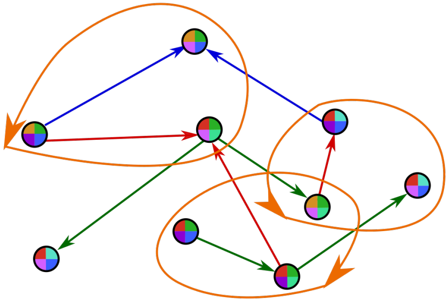


Graph: (V, E)

Attributed graph: $(V, E, \boldsymbol{A})$. Node attributes $\boldsymbol{A}$: *Boolean*, *categorical*, or *numeric*

Attributed multirelational graph: $(V, \boldsymbol{E}, \boldsymbol{A})$. $\boldsymbol{E}$: set of different edge relations

Attributed multirelational hyper-graph: $(V, \boldsymbol{R})$. $\boldsymbol{R}$: set of 1,2,3,...-ary relations (subsumes $\boldsymbol{A}$, $\boldsymbol{E}$)



Graph: (V, E)

Attributed graph: $(V, E, \boldsymbol{A})$. Node attributes $\boldsymbol{A}$: *Boolean*, *categorical*, or *numeric*

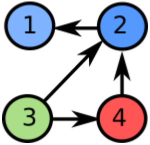
Attributed multirelational graph: $(V, \boldsymbol{E}, \boldsymbol{A})$. $\boldsymbol{E}$: set of different edge relations

Attributed multirelational hyper-graph: $(V, \boldsymbol{R})$. $\boldsymbol{R}$: set of 1,2,3,...-ary relations (subsumes $\boldsymbol{A}$, $\boldsymbol{E}$)

Examples for higher arity relations (logic, relational databases):

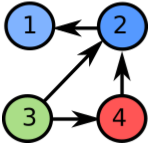
3-ary traffic network relation: *on_shortest_path*(location,location,location)
3-ary movie data: *made_contract*(agent,actor,movie)

Node	color
1	blue
2	blue
3	green
4	red

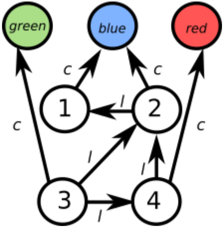


(a)

Node	c_blue	c_green	c_red
1	1	0	0
2	1	0	0
3	0	1	0
4	0	0	1



(b)

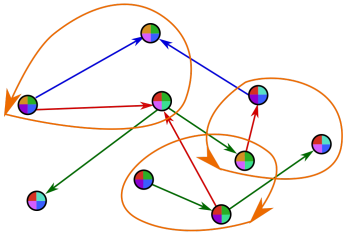


(c)

(a): unary, categorical values

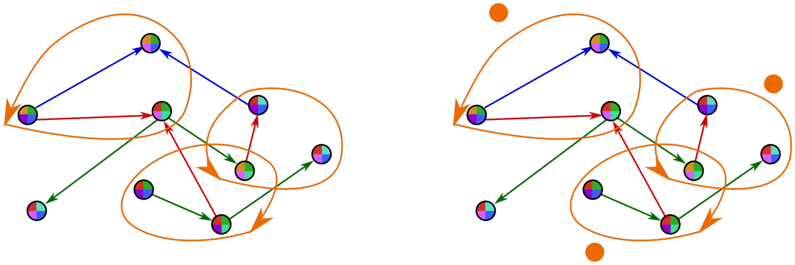
(b): unary, Boolean/binary values (one-hot encoding)

(c): binary relation between objects and attribute values materialized as nodes (example: knowledge graphs)



(a)

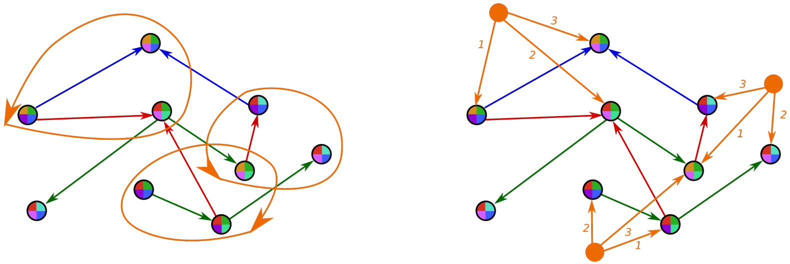
(a): as tuples of nodes



(a) (b)

(a): as tuples of nodes

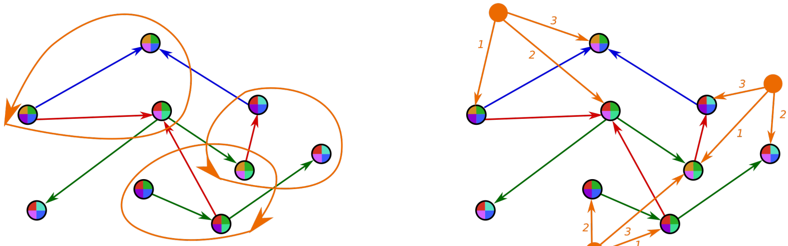
(b): materialize tuples of nodes;



(a) (b)

(a): as tuples of nodes

(b): materialize tuples of nodes;
connect tuple-nodes with entity-nodes by binary relations



(a) (b)

(a): as tuples of nodes

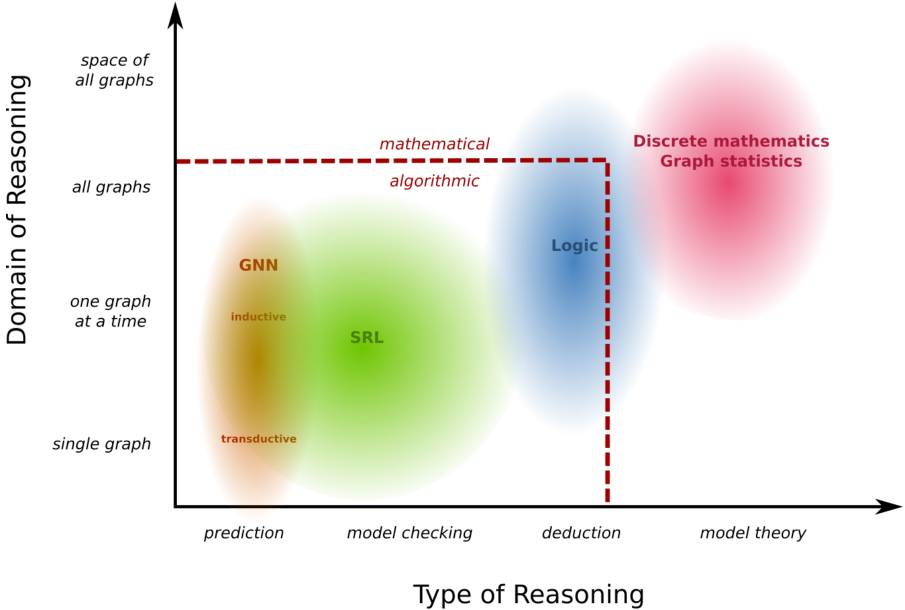
(b): materialize tuples of nodes;
connect tuple-nodes with entity-nodes by binary relations

➥Categorical attributes and relations of higher arities can be reduced to Boolean attributes (one-hot-encodings) and binary relations (but this can be user-unfriendly).

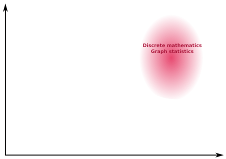
N	number of nodes/vertices
$\mathcal{R}$	a *signature* of 1,2,3,...-ary relation symbols
$\boldsymbol{R}$	specific values of the relations in $\mathcal{R}$ in a graph $G = (V, \boldsymbol{R})$.
$G = (V, \boldsymbol{R})$	a graph with node set V, and relations $\boldsymbol{R}$
$\mathcal{G}(V, \mathcal{R})$	set of all graphs with node set V, and relations in the signature $\mathcal{R}$
$\Delta\mathcal{G}(V, \mathcal{R})$	set of all probability distributions over $\mathcal{G}(V, \mathcal{R})$

Generally assume that $V = \{1, \ldots, N\}$, and $i, j, \ldots \in \mathbb{N}$ denote nodes.

Reasoning



Domain of Reasoning
space of all graphs
all graphs
one graph at a time
single graph
mathematical
algorithmic
GNN
inductive
transductive
SRL
Logic
Discrete mathematics
Graph statistics
prediction
model checking
deduction
model theory
Type of Reasoning

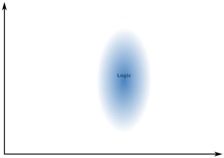


Given: a probabilistic model for the random generation/evolution of graphs.

Question: what is the probability that the graph becomes (stays) connected, as the number of nodes goes to infinity?

➥Or many other questions about the global properties of a random graph model.

➥Mostly (human powered) mathematics, not algorithmic reasoning



Given: a knowledge base

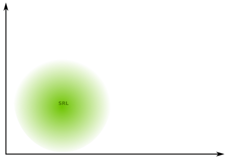
$$\forall x \exists y\ follows(x, y)$$
$$\exists y \neg \exists x\ follows(x, y)$$

Question: Does the knowledge base imply a given query statement?

$$(\exists y \exists^{\geq 2} x\ follows(x, y)) \vee \exists^{\geq 10.000} x\ ?$$

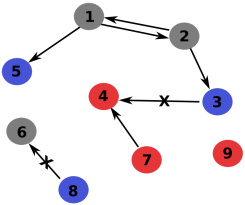
➥Reasoning about all possible graphs

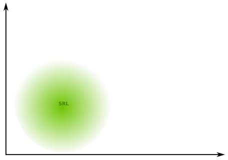
➥Algorithmic reasoning implemented by *theorem provers*.



Given: a generative probabilistic model for graphs.

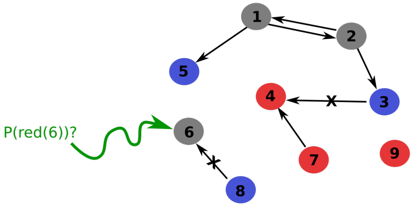
Question: for a single partially observed graph, what are the probabilties of unobserved features?

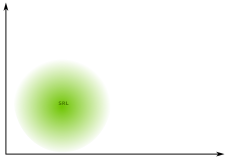




Given: a generative probabilistic model for graphs.

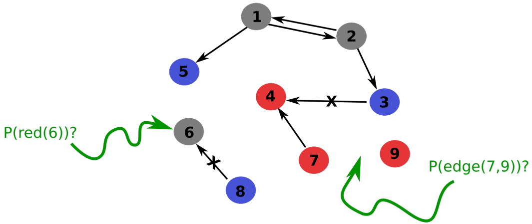
Question: for a single partially observed graph, what are the probabilties of unobserved features?

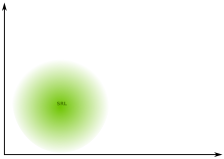




Given: a generative probabilistic model for graphs.

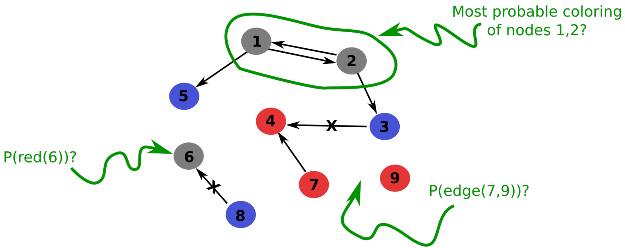
Question: for a single partially observed graph, what are the probabilties of unobserved features?

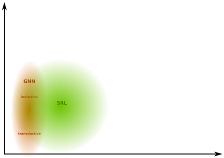




Given: a generative probabilistic model for graphs.

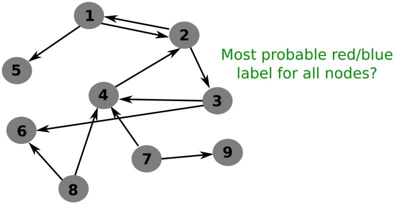
Question: for a single partially observed graph, what are the probabilties of unobserved features?





Given: a descriminative model for specific node label.

Question: for an input graph (edges, node attributes), what are predicted node labels?

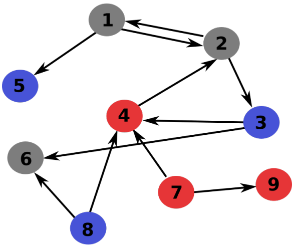


➥Similarly: link prediction, graph classification.

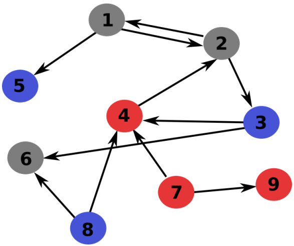
Learning

Learning and Reasoning about a single graph:

Training data

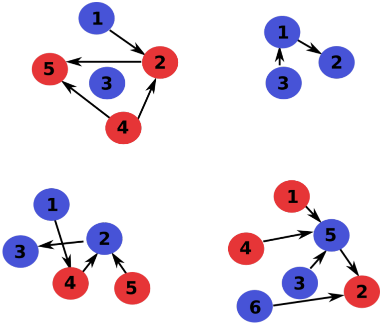


Reasoning domain

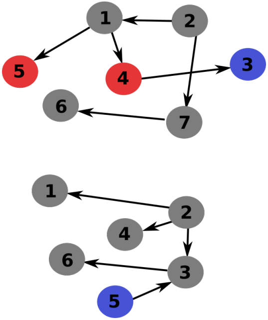


Learning and Reasoning about different graphs:

Training data



Reasoning domains (a.k.a. test cases)



Graph Neural Networks: Basics

$\boldsymbol{h}^k(i)$: d^k-dimensional vector representation of node i at kth iteration (layer).

A basic form of message passing updates:

$$\boldsymbol{h}^0(i) = \text{ initial node feature vector of node } i$$
$$\boldsymbol{h}^{k+1}(i) = f\left(\boldsymbol{W}^k \boldsymbol{h}^k(i) + \boldsymbol{U}^k \sum_{j \in N_i} \boldsymbol{h}^k(j)\right)$$

with ingredients:

- $\boldsymbol{W}^k, \boldsymbol{U}^k$: weight matrices (dimensions: $d^{k+1} \times d^k$)
- f: (nonlinear) activation function (component-wise)

$\boldsymbol{h}^k(i)$: d^k-dimensional vector representation of node i at kth iteration (layer).

A basic form of message passing updates:

$$\boldsymbol{h}^0(i) = \text{ initial node feature vector of node } i$$
$$\boldsymbol{h}^{k+1}(i) = f\left(\boldsymbol{W}^k \boldsymbol{h}^k(i) + \boldsymbol{U}^k \sum_{j \in N_i} \boldsymbol{h}^k(j)\right)$$

with ingredients:

- $\boldsymbol{W}^k, \boldsymbol{U}^k$: weight matrices (dimensions: $d^{k+1} \times d^k$)
- f: (nonlinear) activation function (component-wise)

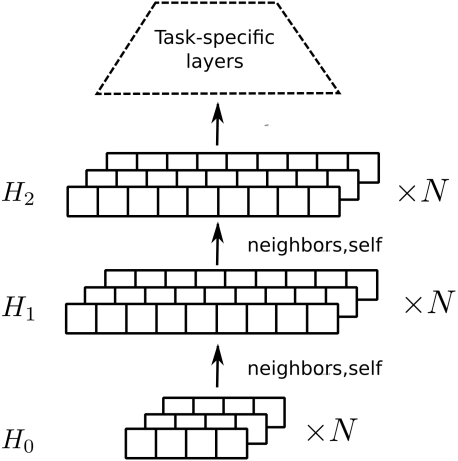
In full matrix notation:

$$\boldsymbol{H}^{k+1} = f\left(\boldsymbol{H}^k(\boldsymbol{W}^k)^T + \boldsymbol{E}\boldsymbol{H}^k(\boldsymbol{U}^k)^T\right)$$

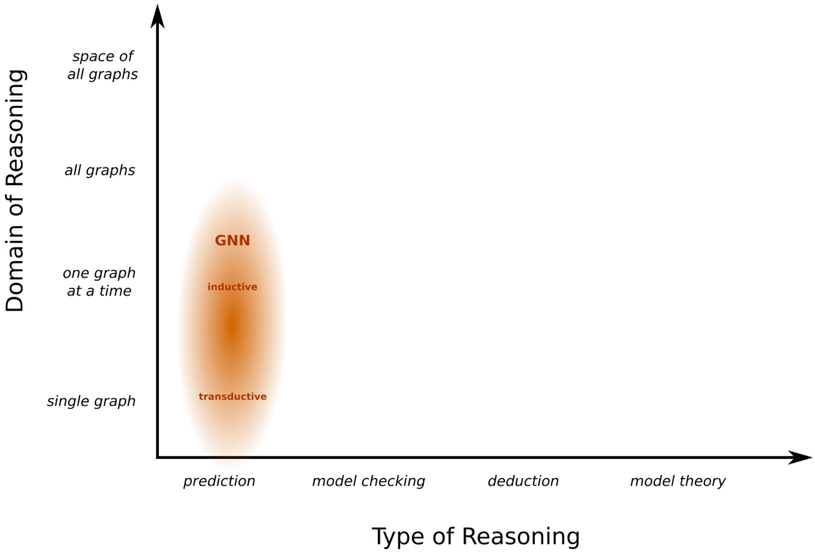
with ingredients:

- $\boldsymbol{H}^k, \boldsymbol{H}^{k+1}$: $n \times d^k$ and $n \times d^{k+1}$ matrices
- $\boldsymbol{E}$: $n \times n$ adjacency matrix

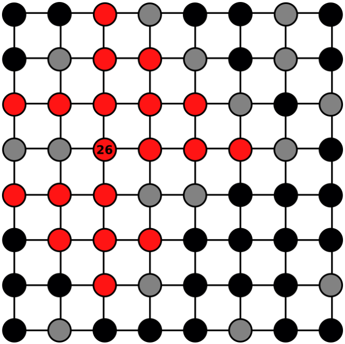
Representation as NN architecture/computation graph:



- At each layer: one vector for each node (picture: $N = 3$)
- At top: task-specific (node or graph classification) transformations of final node representations
- self, neighbors: dependence of vectors in following layer on previous layer


Domain of Reasoning
space of all graphs
all graphs
one graph at a time
single graph
GNN
inductive
transductive
prediction
model checking
deduction
model theory
Type of Reasoning

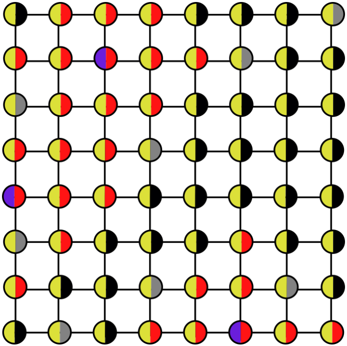
Initial features: node identifiers (typically: one-hot encoded).



Can represent/learn classification rule: node is *red*, if it has distance ≤ 3 to node 26.

➥this only works in transductive settings.

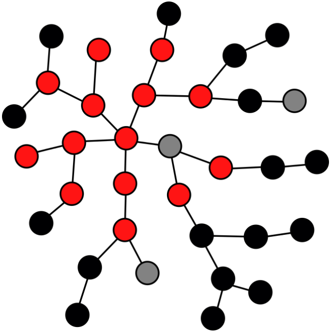
Initial features: node attributes (e.g. *color* $\in \{yellow, blue\}$)



Can represent/learn classification rule: node is *red*, if it has distance ≤ 2 to a blue node.

➥this works in inductive settings: rule can be applied to new graphs with yellow/blue nodes.

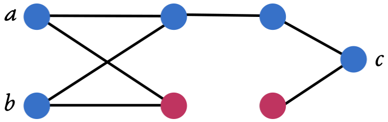
Initial features: none (then can say e.g.: $\boldsymbol{h}^0(i) = 1$ for all i).



Can represent/learn classification rule: node is *red*, if it has distance ≤ 2 to a node with degree ≥ 5.

➡this works in inductive settings: rule can be applied to new graphs.

Discriminative power: when can two nodes be distinguished by a GNN?



a, b indistinguishable by any GNN.

a, c indistinguishable by 2-layer GNNs, distinguishable by 3-layer GNNs.

➥ l-layer GNNs can only access information in the $l - 1$ hop node neighborhood.

- GNNs cannot access "global" graph properties. Examples: cannot recognize whether a graph is connected/disconnected
- GNNs cannot reason about "identity" of nodes (unless node identifiers provided as initial features). Example: cannot recognize whether a node is a member of a clique of size ≥ 3.

Main theorem of [Barceló et al.]:

Every node property that can be expressed in **the two-variable fragment of first-order logic with counting quantifiers** *(FOC_2) can be captured by an ACR-GNN.*

Example

In FOC_2:

$$\alpha_1(X) \equiv \exists^{[8,10]} Y(blue(Y) \wedge \neg edge(X, Y))$$

("there exist 8-10 blue nodes that are not neighbors of *X*")

Not in FOC_2:

$$\delta(X) \equiv \exists Y, Z(X \neq Y \wedge X \neq Z \wedge Y \neq Z \wedge edge(X, Y) \wedge edge(X, Z) \wedge edge(Y, Z))$$

(" *X* is part of a triangle ")

[Barceló, Pablo, et al. "The logical expressiveness of graph neural networks." 2020]

Result from [Jaeger, Relational Bayesian Networks, 1997]:

Let $\phi(\boldsymbol{x})$ be a first-order formula over signature $\mathcal{R}$. Then there exists a probability formula $F_\phi(\boldsymbol{x})$ over $\mathcal{R}$, s.t. for every multi-relational graph $G = (V, \boldsymbol{R})$ and every $|\boldsymbol{x}|$-tuple $\boldsymbol{v}$ of nodes: $F_\phi(\boldsymbol{v}) = 1$ iff $\phi(\boldsymbol{v})$ holds in G (and $F_\phi(\boldsymbol{d}) = 0$ otherwise).

Statistical Relational Learning

An SRL *framework* consists of

- **Syntax:** a formal representation language over relational signatures $\mathcal{R}$
- **Semantics:** defines for any domain V, a probability distribution over the space $\mathcal{G}(V,\mathcal{R})$; formally: a mapping
$$V \mapsto P_V \in \Delta\mathcal{G}(V,\mathcal{R})$$
- **Inference (reasoning):** algorithms for the computation of *conditional probabilities*
$$P_V(A|B) \text{ for some } A, B \subseteq \mathcal{G}(V,\mathcal{R})$$
Also: computing *most probable explanation (MPE)*:
$$\max_{G \in \mathcal{G}(V,\mathcal{R})} P_V(G|B)$$
- **Learning:** methods for learning models from graph (relational) data. Typically divided into:
  - Structure learning: determines (logical) structure of the model (here also: knowledge-driven design)
  - Parameter learning: fitting numerical parameters

Representatives for main paradigms:

RBN	Directed probabilistic graphical models
MLN	Undirected probabilistic graphical models
ProbLog	(Inductive) logic programming

Relational Bayesian Networks

Chain Rule

For fixed V, P_V is a distribution over values $\boldsymbol{R} = (R_1, \ldots, R_r)$. Let $R_{1:h} := (R_1, \ldots, R_h)$. This distribution can be factored as

$$P_V(\boldsymbol{R}) = P_V(R_1) \cdot P_V(R_2|R_1) \cdot \ldots \cdot P_V(R_h|R_{1:h-1}) \cdot \ldots \cdot P_V(R_r|R_{1,r-1}).$$

Conditional independence of relations

Conditional independencies lead to simplifications:

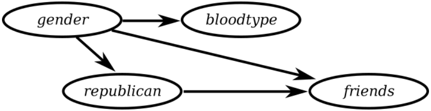
$$P_V(R_h|R_{1:h-1}) = P_V(R_h|Pa(R_h)) \text{ for some } Pa(R_h) \subset R_{1:h-1}$$

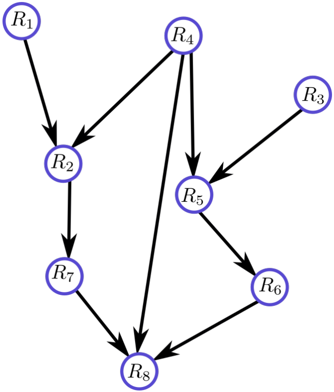
➥directed acyclic graph over relations (*relation DAG*).

$P_V(gender, republican, bloodtype, friends) =$

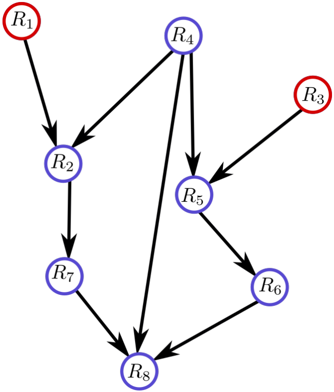
$P_V(gender)P_V(republican|gender)P_V(bloodtype|republican, gender)P_V(friends|bloodtype, republican, gender) \overset{\text{assume}}{=}$

$P_V(gender)P_V(republican|gender)P_V(bloodtype|gender)P_V(friends|republican, gender)$





- Defines full generative probabilistic model for graphs in signature $\mathcal{R}$

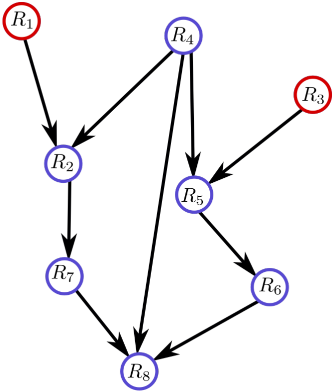


- Defines full generative probabilistic model for graphs in signature $\mathcal{R}$
- Sometimes: assume some relations $R \in \mathcal{R}$ are predefined input relations:

$$\mathcal{R} = \mathcal{R}_{prob} \cup \mathcal{R}_{in}$$

  - make these relations roots in the relation DAG
  - do not define a distribution $P_V(R_h)$ for these relations
  - defines a *conditional* distribution

$$P_V(\boldsymbol{R}_{prob} | \boldsymbol{R}_{in})$$



- Defines full generative probabilistic model for graphs in signature $\mathcal{R}$
- Sometimes: assume some relations $R \in \mathcal{R}$ are predefined input relations:

$$\mathcal{R} = \mathcal{R}_{prob} \cup \mathcal{R}_{in}$$

  - make these relations roots in the relation DAG
  - do not define a distribution $P_V(R_h)$ for these relations
  - defines a *conditional* distribution

$$P_V(\mathbf{R}_{prob} | \mathbf{R}_{in})$$

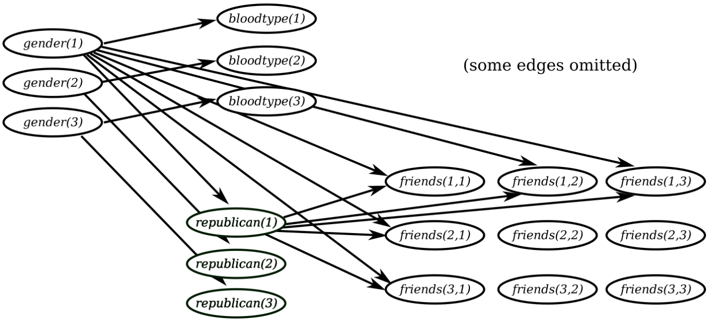
➥All SRL frameworks support divisions $\mathcal{R} = \mathcal{R}_{prob} \cup \mathcal{R}_{in}$

Atom independence

Assume atoms of one relation are mutually independent, given the parent relations:

$$P_V(R_h|Pa(R_h)) := \prod_{\boldsymbol{i}\in V^{arity(R_h)}} P_V(R_h(\boldsymbol{i})|Pa(R_h))$$

As a Bayesian network:



➥Leads to limitations for modeling e.g. symmetry constaints *friends*(1, 2) ⇔ *friends*(2, 1), or homophily (exist modeling tricks to circumvent this!).

A relational Bayesian network for signature $\mathcal{R}$ consists of

- a directed acyclic graph whose nodes are the relations $R \in \mathcal{R}$,
- for each $R \in \mathcal{R}$ a *probability formula* F_R *in the signature* $Pa(R)$ that defines the conditional probabilities

$$P_V(R(\boldsymbol{i})|Pa(R))$$

Probability formulas: semantics

A probability formula F maps tuples of entities $\boldsymbol{i}$ in a graph $G = (V, \boldsymbol{R})$ to a probability value

$$eval(F, \boldsymbol{i}, G) \in [0, 1]$$

[M. Jaeger: Relational Bayesian Networks. UAI 1997]

Constants

For any $q \in [0, 1]$,

$$F \equiv q$$

is a probability formula with

$$eval(F, \boldsymbol{i}, G) = q$$

for all $\boldsymbol{i}, G$.

Example

Let $\mathcal{R} = \{edge\}$. Then

$$F_{edge(X,Y)} \equiv 0.5$$

defines the classic Erdős-Rényi random graph model.

Atoms

For any $R \in \mathcal{R}$, and variables $Y_1, \ldots, Y_{arity(R)}$

$$F \equiv R(Y_1, \ldots, Y_{arity(R)})$$

is a probability formula with

$$eval(F, \boldsymbol{i}, G) = \begin{cases} 1 & \text{if } R(\boldsymbol{i}) \text{ is true in } G \\ 0 & \text{if } R(\boldsymbol{i}) \text{ is false in } G \end{cases}$$

WIF-THEN-ELSE

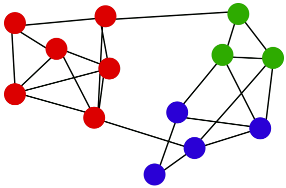
If F_1, F_2, F_3 are probability formulas, then

$$F \equiv \texttt{WIF} \;\; F_1 \;\; \texttt{THEN} \;\; F_2 \;\; \texttt{ELSE} \;\; F_3$$

is a probability formula with

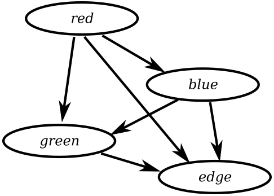
$$eval(F, \boldsymbol{i}, G) = eval(F_1, \boldsymbol{i}, G) eval(F_2, \boldsymbol{i}, G) + (1 - eval(F_1, \boldsymbol{i}, G)) eval(F_3, \boldsymbol{i}, G)$$

➥Generalization of Boolean operations ($F_i \in \{0, 1\}$)



- Nodes partitioned into *blocks*
- Probability of edges depends on block memberships

With the constructs introduced so far:



A. partitioning into red, green, blue nodes:

$$F_{red(X)} \equiv 0.5$$
$$F_{blue(X)} \equiv \texttt{WIF } red(X) \texttt{ THEN } 0 \texttt{ ELSE } 0.7$$
$$F_{green(X)} \equiv \texttt{WIF } red(X) \vee blue(X) \texttt{ THEN } 0 \texttt{ ELSE } 1.0$$

B. generating edges:

$$F_{edge(X,Y)} \equiv \texttt{WIF } red(X) \wedge red(Y) \texttt{ THEN } 0.6 \texttt{ ELSEIF} \ldots$$

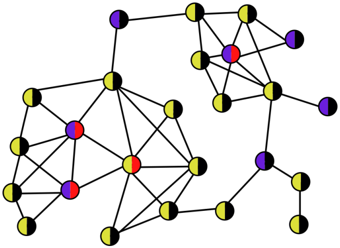
Combination Function

(related to first-order quantifiers ∀, ∃, GNN message passing aggregation, ...)

If $F_1, \ldots, F_t$ are probability formulas, then

$$F \equiv \quad \texttt{COMBINE}\ F_1, \ldots, F_t$$
$$\texttt{WITH}\ < \textit{combination function} >$$
$$\texttt{FORALL}\ < \textit{variables} >$$
$$\texttt{WHERE}\ < \textit{logical constraint} >$$

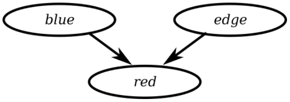
is a probability formula.

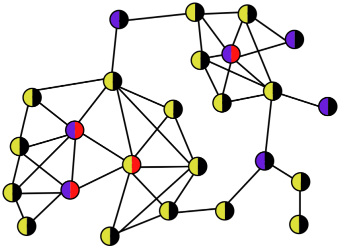


$\mathcal{R} = \{red, blue, edge\}$. In figure: yellow $\sim$ not blue; black $\sim$ not red.

$P_V(red(i))$ higher if

- *i* is *blue*
- *i* is part of many triangles

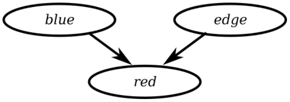




$\mathcal{R} = \{red, blue, edge\}$. In figure: yellow $\sim$ not blue; black $\sim$ not red.

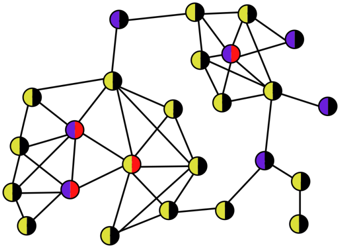
$P_V(red(i))$ higher if

- *i* is *blue*
- *i* is part of many triangles



Defining triangles:

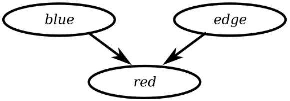
$$F_{triangle(X,Y,Z)} \equiv$$
$$edge(X,Y) \wedge edge(X,Z) \wedge edge(Y,Z)$$



$\mathcal{R} = \{red, blue, edge\}$. In figure: yellow $\sim$ not blue; black $\sim$ not red.

$P_V(red(i))$ higher if

- i is *blue*
- i is part of many triangles



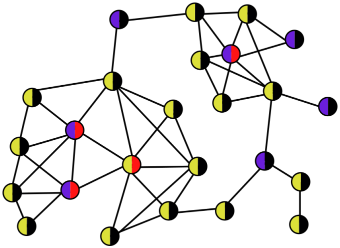
Defining triangles:

$$F_{triangle(X,Y,Z)} \equiv$$
$$edge(X,Y) \wedge edge(X,Z) \wedge edge(Y,Z)$$

Counting triangles:

$$F_{triangle\_count(X)} \equiv$$

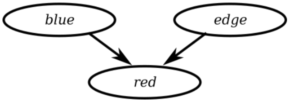
```
COMBINE 1.0
WITH sum
FORALL Y, Z
WHERE F_triangle(X,Y,Z)(X, Y, Z)
```



$\mathcal{R} = \{red, blue, edge\}$. In figure: yellow $\sim$ not blue; black $\sim$ not red.

$P_V(red(i))$ higher if

- i is *blue*
- i is part of many triangles



Defining triangles:

$$F_{triangle(X,Y,Z)} \equiv$$
$$edge(X,Y) \wedge edge(X,Z) \wedge edge(Y,Z)$$

Counting triangles:

$$F_{triangle\_count(X)} \equiv$$

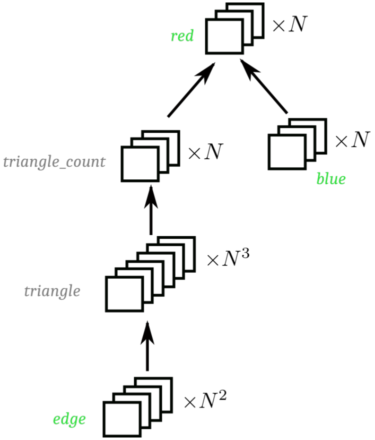
```
COMBINE 1.0
WITH sum
FORALL Y, Z
WHERE F_triangle(X,Y,Z)(X, Y, Z)
```

Logistic regression of *triangle_count* and *blue* feature:

$$F_{red(X)} \equiv$$

```
COMBINE 0.6 · F_triangle_count(X)(X),
        0.3 · blue(X),
        −3.0
WITH logistic regression
```

The computation graph of the probability formula for *red*. In green: relations from $\mathcal{R}$. In gray: synthetic names for intermediate formulas ("layers").



- Each probability (sub-)formula defines a feature of $0, 1, 2, \ldots$-tuples of entities
- Nested formulas give "deep" models
- Aggregation (message passing) along "channels" defined by the constraints in combination functions
- Scalar features

GNN-2-RBN

From [Barceló et al.,2020]:

Input graphs defined by signature:

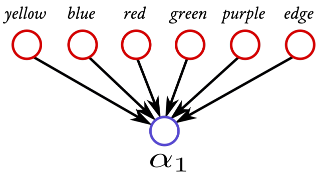
$$\mathcal{R}_{in} = \{blue, green, red, yellow, purple, edge\}$$

Target concept to represent/learn:

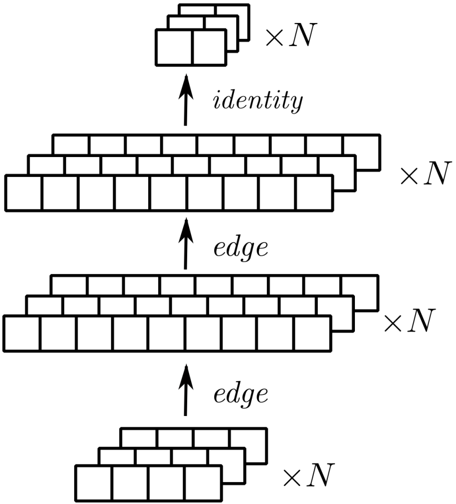
$$\alpha_1(X) \equiv \exists^{[8,10]} Y(blue(Y) \land \neg edge(X,Y))$$

(cf. slide 22)

A GNN α_1 classifier defines a conditional distribution $P(\alpha_1 | blue, green, red, yellow, purple, edge)$ satisfying the *Atom Independence* property (slide 28).

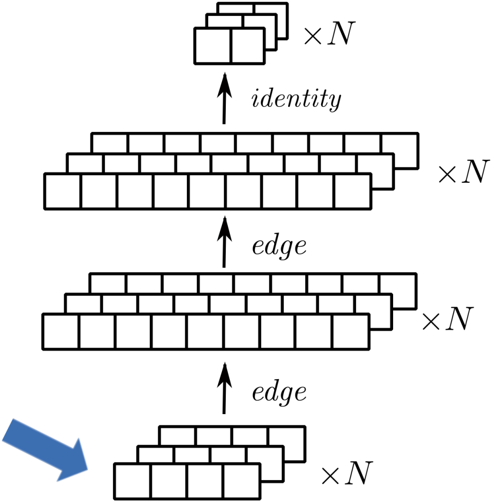


➥This distribution can be encoded by a probability formula.


×N
identity
×N
edge
×N
edge
×N

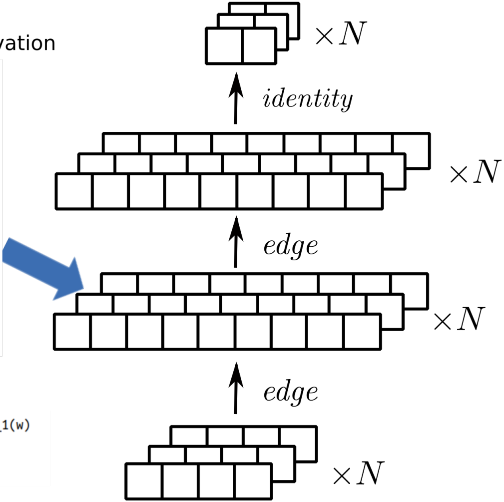
Node attributes

```
@layer_0_0([node]v)= blue(v);
@layer_0_1([node]v)= green(v);
@layer_0_2([node]v)= red(v);
@layer_0_3([node]v)= yellow(v);
@layer_0_4([node]v)= purple(v);
```



Linear combination and activation

```
@layer_1_0([node]v)= COMBINE
         ($c_1_0_0*@layer_0_0(v)),
         ($c_1_0_1*@layer_0_1(v)),
         ($c_1_0_2*@layer_0_2(v)),
         ($c_1_0_3*@layer_0_3(v)),
         ($c_1_0_4*@layer_0_4(v)),
         ($A_1_0_0*@agg_0_0(v)),
         ($A_1_0_1*@agg_0_1(v)),
         ($A_1_0_2*@agg_0_2(v)),
         ($A_1_0_3*@agg_0_3(v)),
         ($A_1_0_4*@agg_0_4(v)),
         ($R_1_0_0*@read_0_0()),
         ($R_1_0_1*@read_0_1()),
         ($R_1_0_2*@read_0_2()),
         ($R_1_0_3*@read_0_3()),
         ($R_1_0_4*@read_0_4()),
         $b_1_0
         WITH l-reg
```



Aggregation

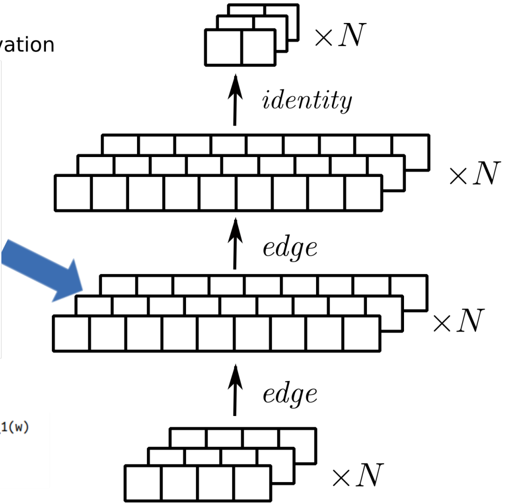
```
@agg_0_1([node]v) = COMBINE @layer_0_1(w)
        WITH sum
        FORALL w
        WHERE (edge(v,w)|edge(w,v));
```

Linear combination and activation

```
@layer_1_0([node]v)= COMBINE
          ($c_1_0_0*@layer_0_0(v)),
          ($c_1_0_1*@layer_0_1(v)),
          ($c_1_0_2*@layer_0_2(v)),
          ($c_1_0_3*@layer_0_3(v)),
          ($c_1_0_4*@layer_0_4(v)),
          ($A_1_0_0*@agg_0_0(v)),
          ($A_1_0_1*@agg_0_1(v)),
          ($A_1_0_2*@agg_0_2(v)),
          ($A_1_0_3*@agg_0_3(v)),
          ($A_1_0_4*@agg_0_4(v)),
          ($R_1_0_0*@read_0_0()),
          ($R_1_0_1*@read_0_1()),
          ($R_1_0_2*@read_0_2()),
          ($R_1_0_3*@read_0_3()),
          ($R_1_0_4*@read_0_4()),
          $b_1_0
          WITH l-reg
```

Aggregation

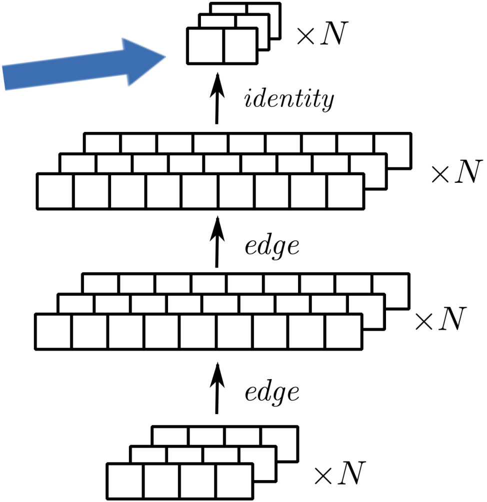
```
@agg_0_1([node]v) = COMBINE @layer_0_1(w)
         WITH sum
         FORALL w
         WHERE (edge(v,w)|edge(w,v));
```



Yellow highlight: trainable parameters $\sim$ entries of GNN weight matrices

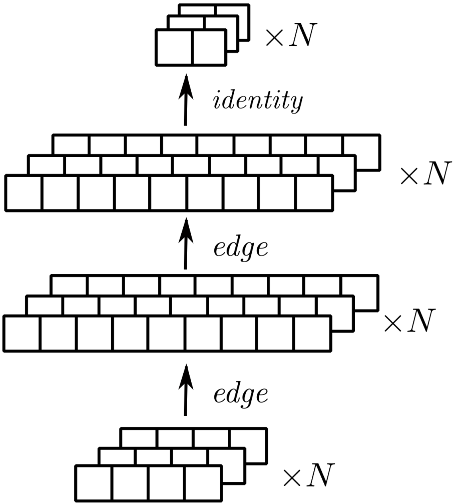
Linear classifier

```
alpha1([node]v)= COMBINE
          ($w_0*@layer_1_0(v)),
          ($w_1*@layer_1_1(v)),
          ($w_2*@layer_1_2(v)),
          ($w_3*@layer_1_3(v)),
          $w_5
          WITH l-reg
```



Yellow highlight: trainable parameters $\sim$ entries of GNN weight matrices

- One-to-one mapping of representation and parameterization
- Matrix-vector level GNN specifications broken down to the "scalar" level
- GNN training $\sim$ RBN learning (same objective, same gradients, . . .)

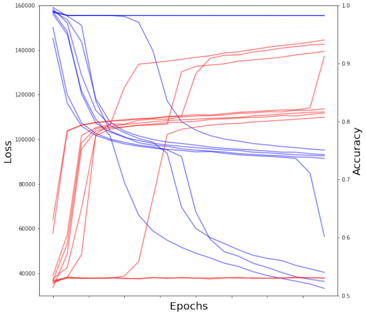


Yellow highlight: trainable parameters $\sim$ entries of GNN weight matrices

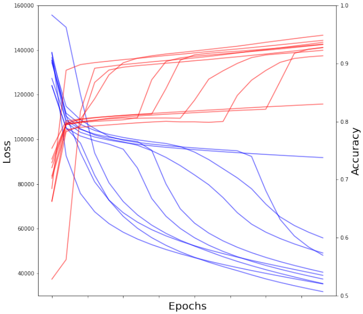
Learning the α_1 target.

Training data: 5000 random graphs of size $N \in 40..50$ (data from [Barceló et al.]).

Pytorch geometric implementation of ACR-GNN:



Primula implementation of RBN encoding:

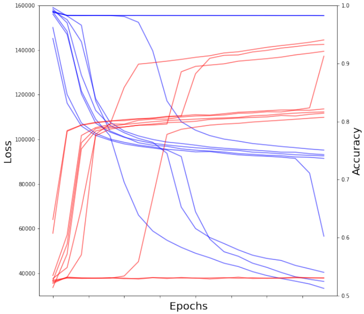


(blue: loss, red: accuracy (on training data); 20 epochs, 10 restarts with random parameter initializations)

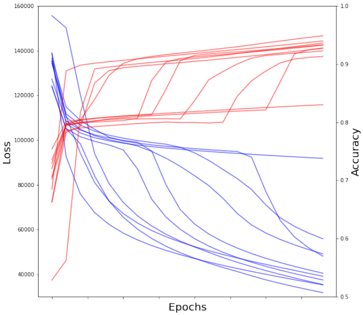
Learning the α_1 target.

Training data: 5000 random graphs of size $N \in 40..50$ (data from [Barceló et al.]).

Pytorch geometric implementation of ACR-GNN:



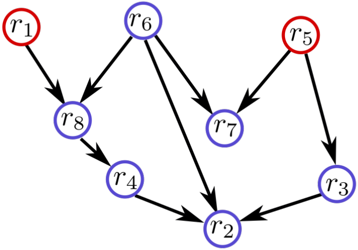
Primula implementation of RBN encoding:



(blue: loss, red: accuracy (on training data); 20 epochs, 10 restarts with random parameter initializations)

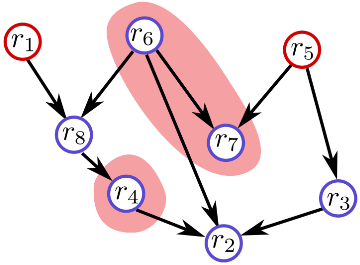
But: Primula takes **much** longer ...

Building a conditional generative model:



Determine relational dependencies and input relations

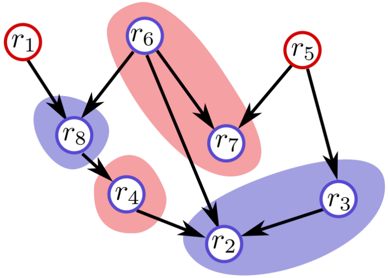
Building a conditional generative model:



Determine relational dependencies and input relations

Rich data, little knowledge: use GNN modules (many parameters, little structure) to define conditional distributions

Building a conditional generative model:

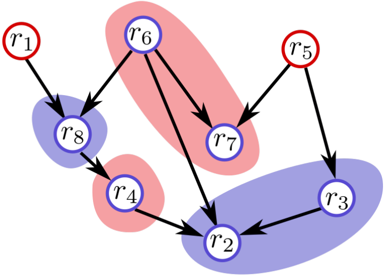


Determine relational dependencies and input relations

Rich data, little knowledge: use GNN modules (many parameters, little structure) to define conditional distributions

Sparse data, expert knowledge, constraints: use customized probability formula (few parameters, highly structured) to define conditional distributions

Building a conditional generative model:



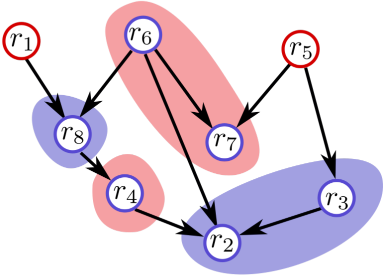
Determine relational dependencies and input relations

Rich data, little knowledge: use GNN modules (many parameters, little structure) to define conditional distributions

Sparse data, expert knowledge, constraints: use customized probability formula (few parameters, highly structured) to define conditional distributions

➥No a-priori distinction of low-level perceptual vs. high-level cognitive reasoning (cf. DeepProbLog)

Building a conditional generative model:



Determine relational dependencies and input relations

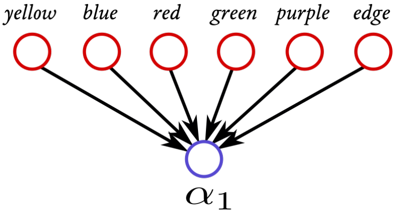
Rich data, little knowledge: use GNN modules (many parameters, little structure) to define conditional distributions

Sparse data, expert knowledge, constraints: use customized probability formula (few parameters, highly structured) to define conditional distributions

➥No a-priori distinction of low-level perceptual vs. high-level cognitive reasoning (cf. DeepProbLog)

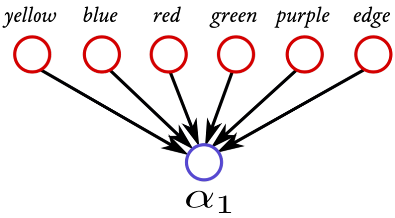
➥The resulting neuro-symbolic model supports all types of model checking reasoning.

Making the α_1 model generative:

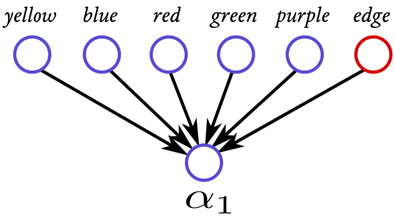


Conditional (prediction) model for α_1 given all other relations as input.

Making the α_1 model generative:



Conditional (prediction) model for α_1 given all other relations as input.



Generative model for node attributes and label, given *edge* as input

$F_{yellow(X)} = 0.18;$

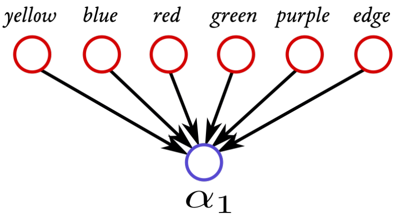
$F_{blue(X)} = 0.26;$

$F_{red(X)} = 0.18;$

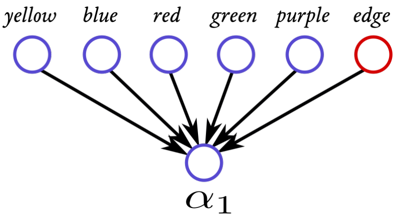
$F_{green(X)} = 0.18;$

$F_{purple(X)} = 0.18;$

Making the α_1 model generative:



Conditional (prediction) model for α_1 given all other relations as input.



Generative model for node attributes and label, given *edge* as input

$F_{yellow(X)} = 0.18;$

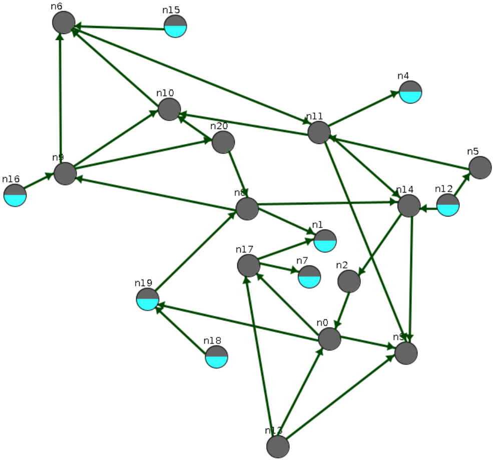
$F_{blue(X)} = 0.26;$

$F_{red(X)} = 0.18;$

$F_{green(X)} = 0.18;$

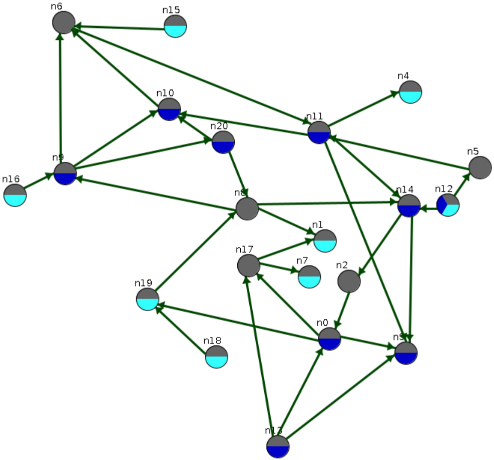
$F_{purple(X)} = 0.18;$

MPE task: given observed α_1 labels, what is the most probable configuration of the *blue* attribute?



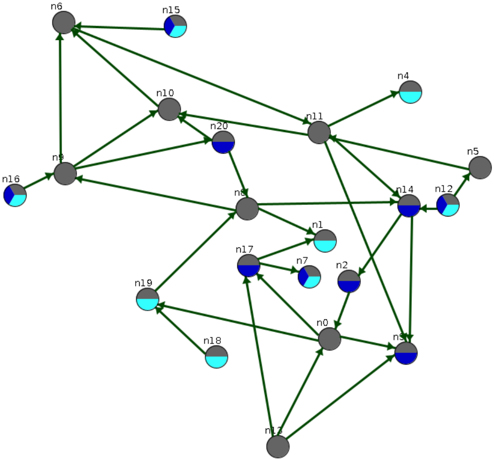
- Graph with observed α_1 relation (21 nodes)

[R. Pojer, A. Passerini, M. Jaeger: Generalized Reasoning with Graph Neural Networks by Relational Bayesian Network Encodings. In Learning on Graphs Conference (2023)].



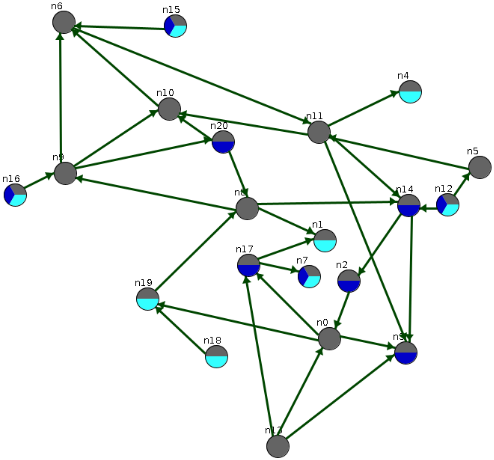
- Graph with observed α_1 relation (21 nodes)
- MAP for *blue* with RBN-GNN manually set parameters (exactly implementing logical definition of α_1; test accuracy: 1.0).

[R. Pojer, A. Passerini, M. Jaeger: Generalized Reasoning with Graph Neural Networks by Relational Bayesian Network Encodings. In Learning on Graphs Conference (2023)].



- Graph with observed α_1 relation (21 nodes)
- MAP for *blue* with RBN-GNN manually set parameters (exactly implementing logical definition of α_1; test accuracy: 1.0).
- MAP for *blue* with RBN-GNN learned parameters (approximately implementing logical definition of α_1; test accuracy: 1.0).

[R. Pojer, A. Passerini, M. Jaeger: Generalized Reasoning with Graph Neural Networks by Relational Bayesian Network Encodings. In Learning on Graphs Conference (2023)].



- Graph with observed α_1 relation (21 nodes)
- MAP for *blue* with RBN-GNN manually set parameters (exactly implementing logical definition of α_1; test accuracy: 1.0).
- MAP for *blue* with RBN-GNN learned parameters (approximately implementing logical definition of α_1; test accuracy: 1.0).

➥perfect accuracy on primary prediction task does not guarantee perfect accuracy for other reasoning tasks.

[R. Pojer, A. Passerini, M. Jaeger: Generalized Reasoning with Graph Neural Networks by Relational Bayesian Network Encodings. In Learning on Graphs Conference (2023)].

- There is more to reasoning than prediction!
- GNNs: good at learning accurate predictors from data
- SRL: good at flexible reasoning
- RBNs: the SRL framework most closely related to GNNs
- RBN+GNN: seamless integration of GNN prediction models into SRL model