IBM Research

Neuro-symbolic AI : where are the rules ?

Christian de Sainte Marie

IBM

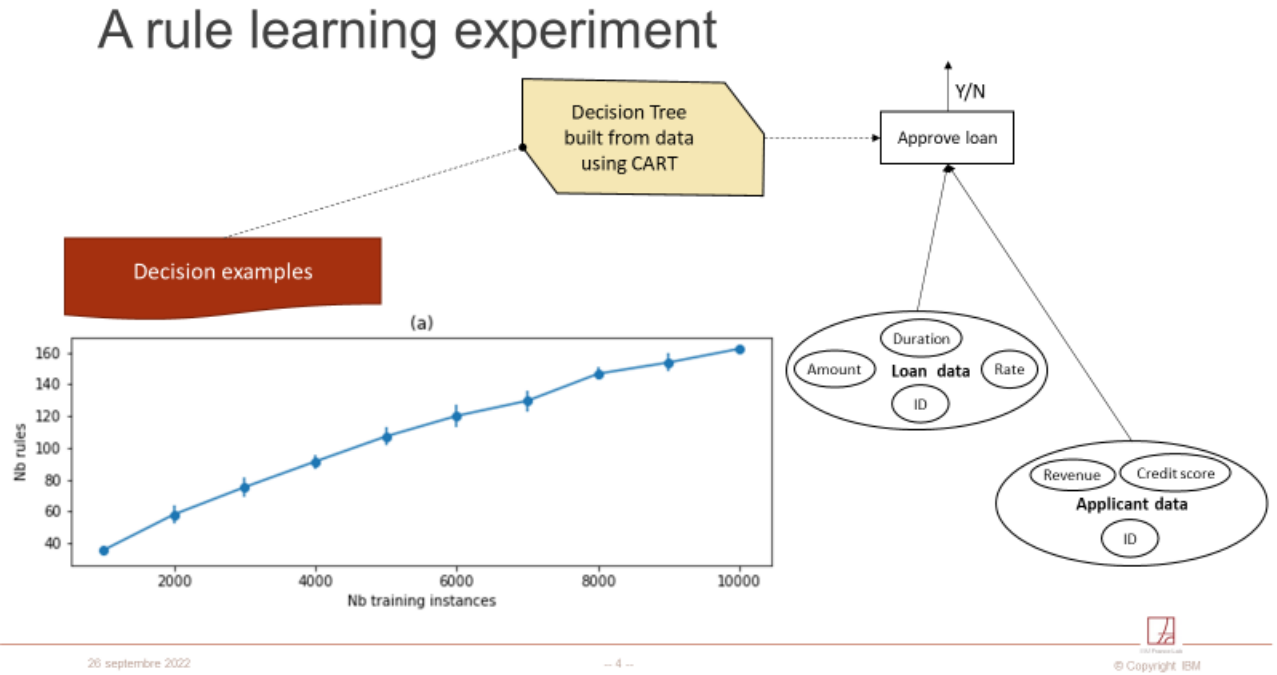
Motivation

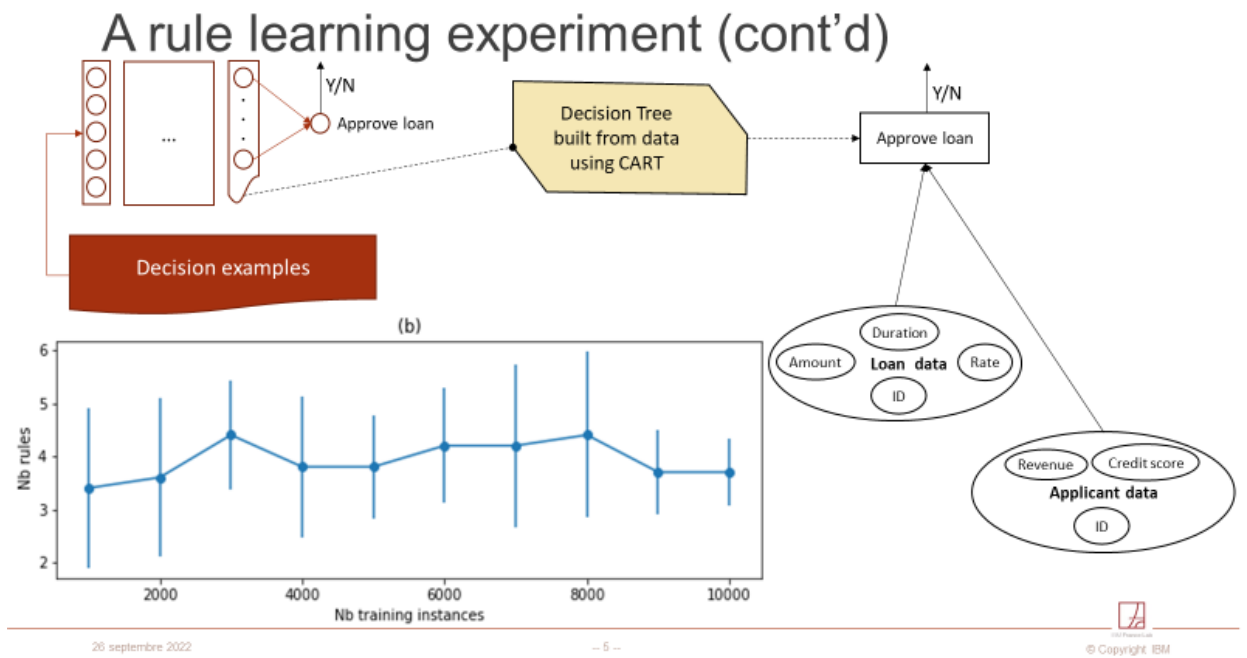
- We need symbolic decision models (rules, theories…)
  - Many automation scenarios require ex ante validation
  - Learning is about getting beyond statistics

- But pure symbolic rule learning methods are not good at capturing the right level of data representation

Motivation

If amount > 1M then approve = No
else If credit score < 200 then approve = No
else if Amount*(1+Rate) / Duration*Revenue _> 0.3 then approve = No
Else approve = Yes





*Christian de Sainte Marie, **Learning decision rules or learning decision models ?**, 5th International Joint Conférence on Rules and Reasoning 2021, LNCS volume 12851, Springer Verlag*

Motivation

- We need symbolic decision models
  - Many automation scenarios require ex ante validation
  - Learning is about getting beyond statistics
- But pure symbolic rule learning methods are not good at capturing the right level of data representation

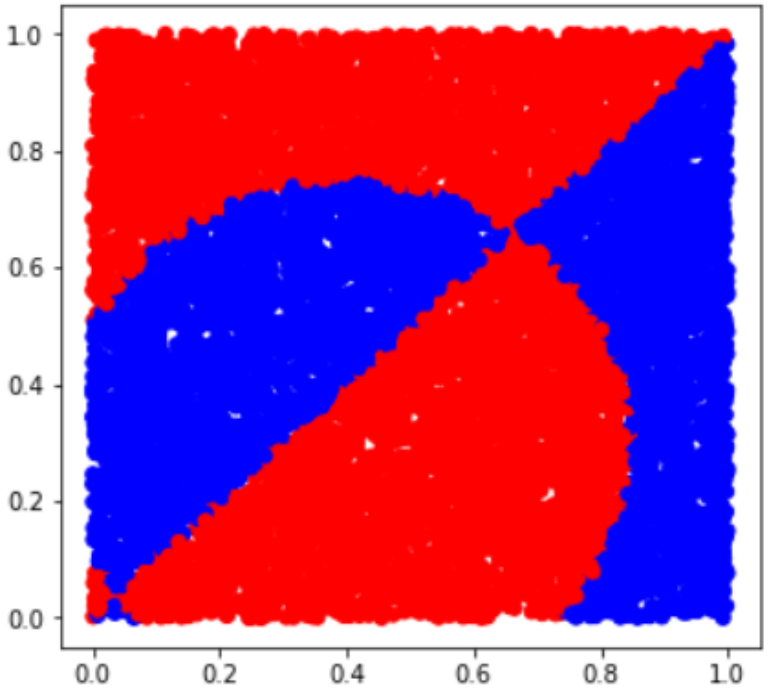
- NN are supposedly good at capturing the right level of data representation
  - But how good are they at capturing rules ?

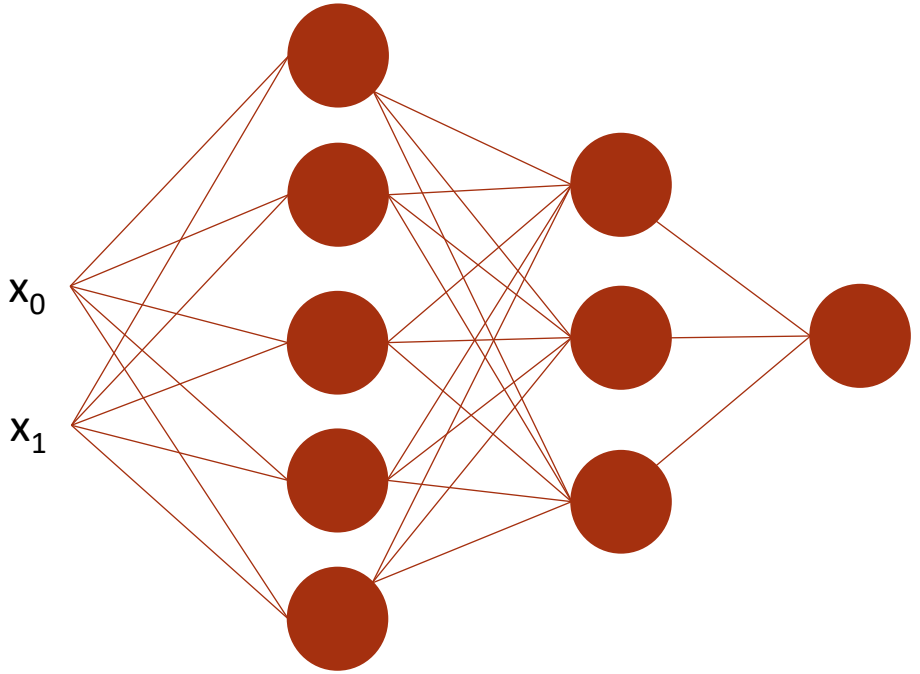
Where are the rules ?

Example

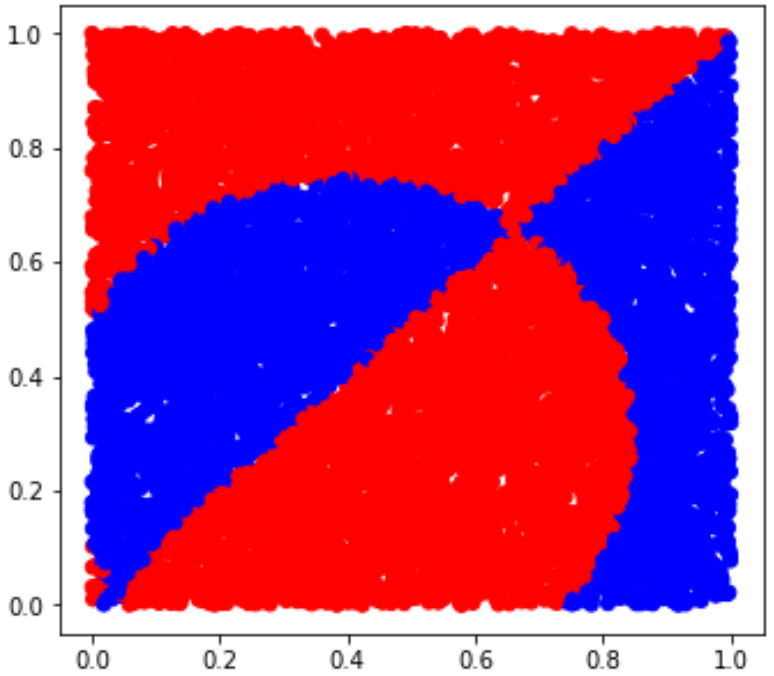
If (the point is in the circle and below the line)
or (the point is outside of the circle and above the line)
then the point is red else it is blue

Training data

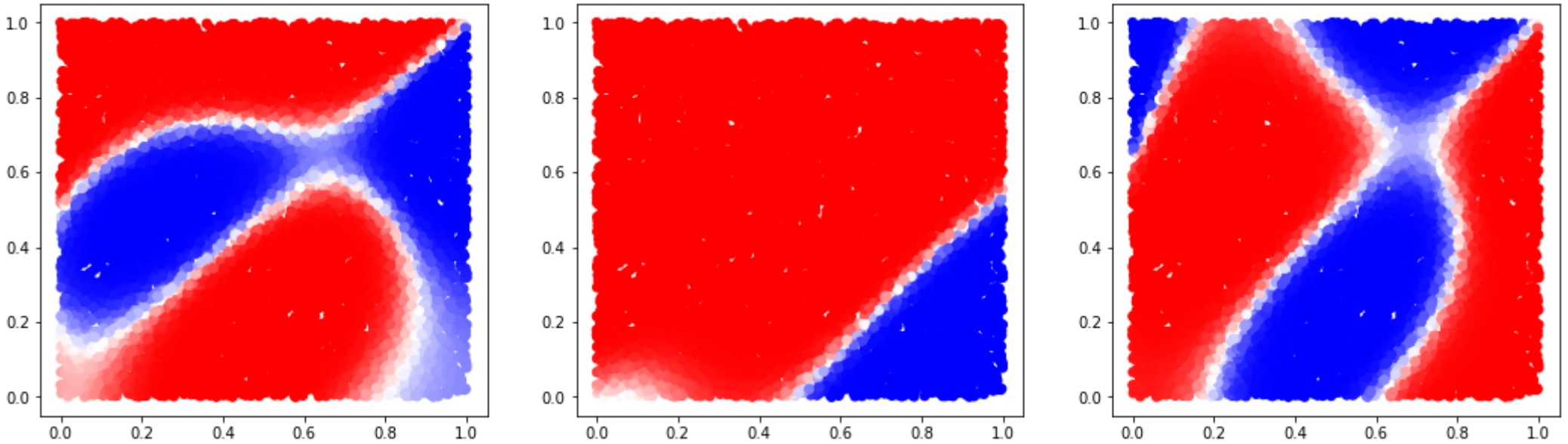




Trained MLP

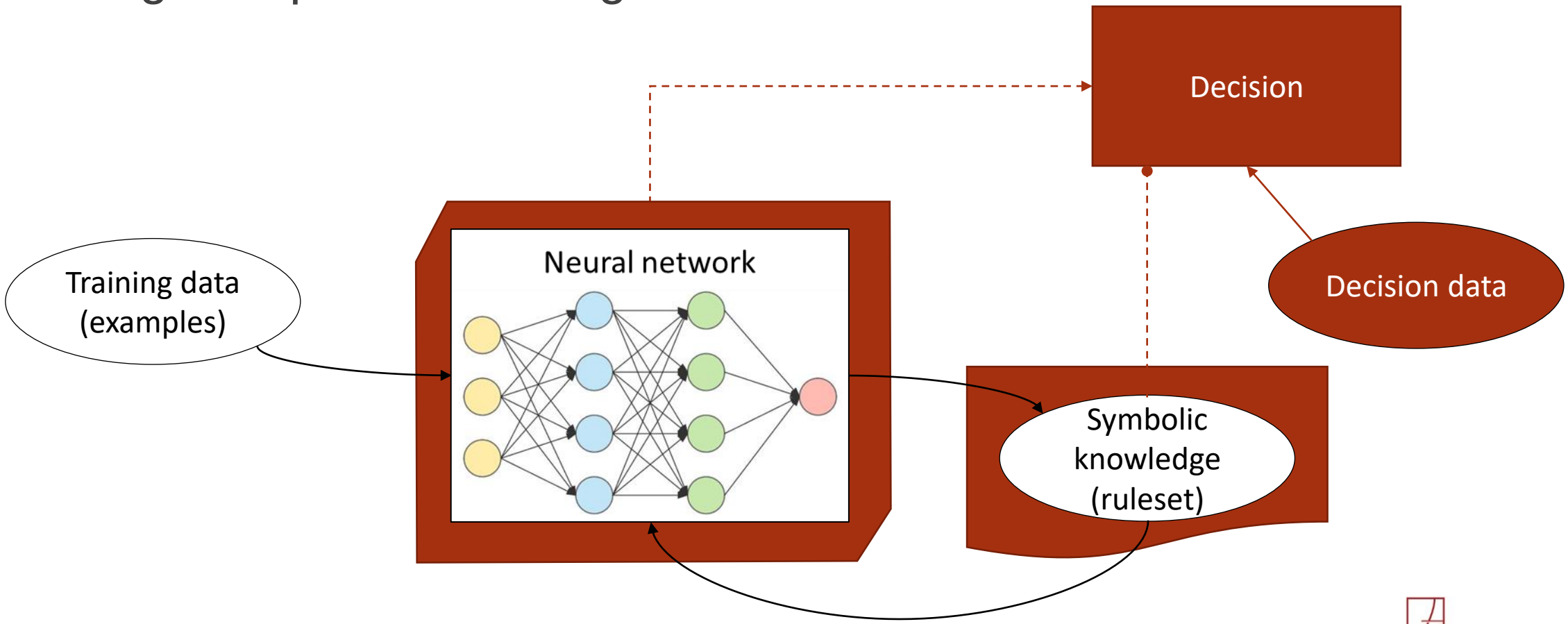


Output of the last hidden layer



Rule injection

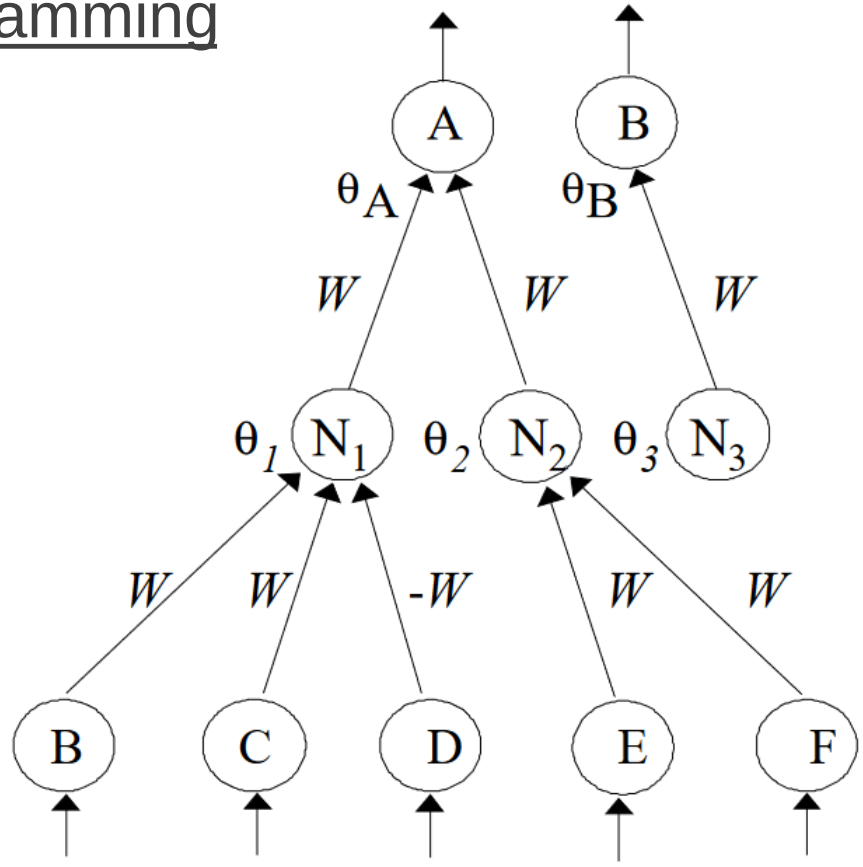
- Inference (parallel execution)
- Training with prior knowledge



C- IL^{2}P

Connectionist Inductive Learning and Logic Programming

- Add an output node for each atom in the head of a clause
  - Compute the threshold θ associated with that node
- Add one node in the hidden layer for each clause
  - Connect each “clause” node to the output node corresponding to its head, with weight *W*
  - Compute the threshold θ associated with that node
- Add an input node for each litteral appearing in the body of a clause
  - Connect each input node to each hidden layer node that corresponds to a clause which body includes the corresponding litteral, with weight *W* if the litteral is positive in the clause, otherwise -*W*



NN representing the ruleset (kernel)

- $B \wedge C \wedge \neg D \rightarrow A$
- $E \wedge F \rightarrow A$
- B

A. d’Avila Garcez, G. Zaverucha, ***The Connectionist Inductive Learning and Logic Programming System****, Applied Intelligence volume 11, pages 59–77 (1999)*

Example

Logic program

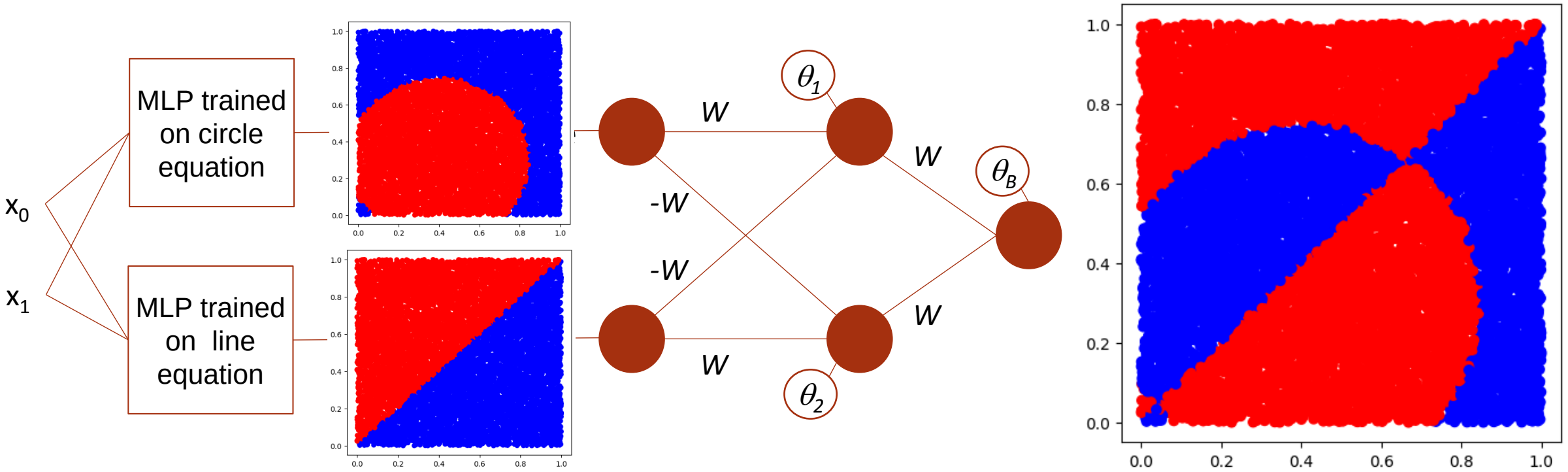
Red :- inCircle $\wedge$ $\neg$belowLine (1)

Red :- $\neg$inCircle $\wedge$ belowLine (2)

(Blue :- $\neg$Red)

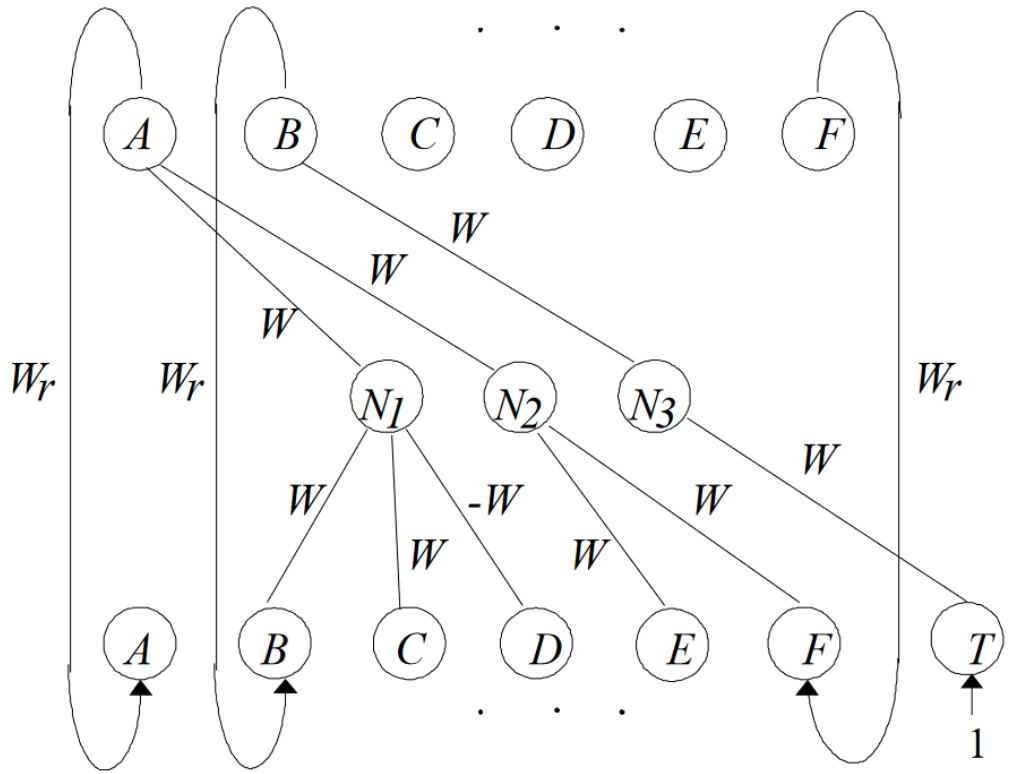
Parameters

- W = 4.4
- $\theta_1 = \theta_2 = \theta_B = -3.3$



C- IL^{2}P and beyond

- **Existence** : *For each propositional general logic program P, a feedforward kernel NN K_P with exactly one hidden layer can be built that computes the immediate consequence operator T_P.*
- **Training** : *add nodes to input and output layers according to the training set, add nodes to the hidden layer for new rules, add connections with null weights (+/- ε), train using gradient backpropagation as usual.*
- **Inference** : *For each acceptable general program P, T_P has a unique fixed point, and the recurrent NN that connects the ouputs of the kernel NN K_P to its inputs with recurrent weights $W_r = 1$ converges for any initial input to that fixed point (which is the stable model).*
- Hitzler et al. extend that result to generalized consequence operators and first-order logic programs (under conditions).



Recurrent NN extending the kernel

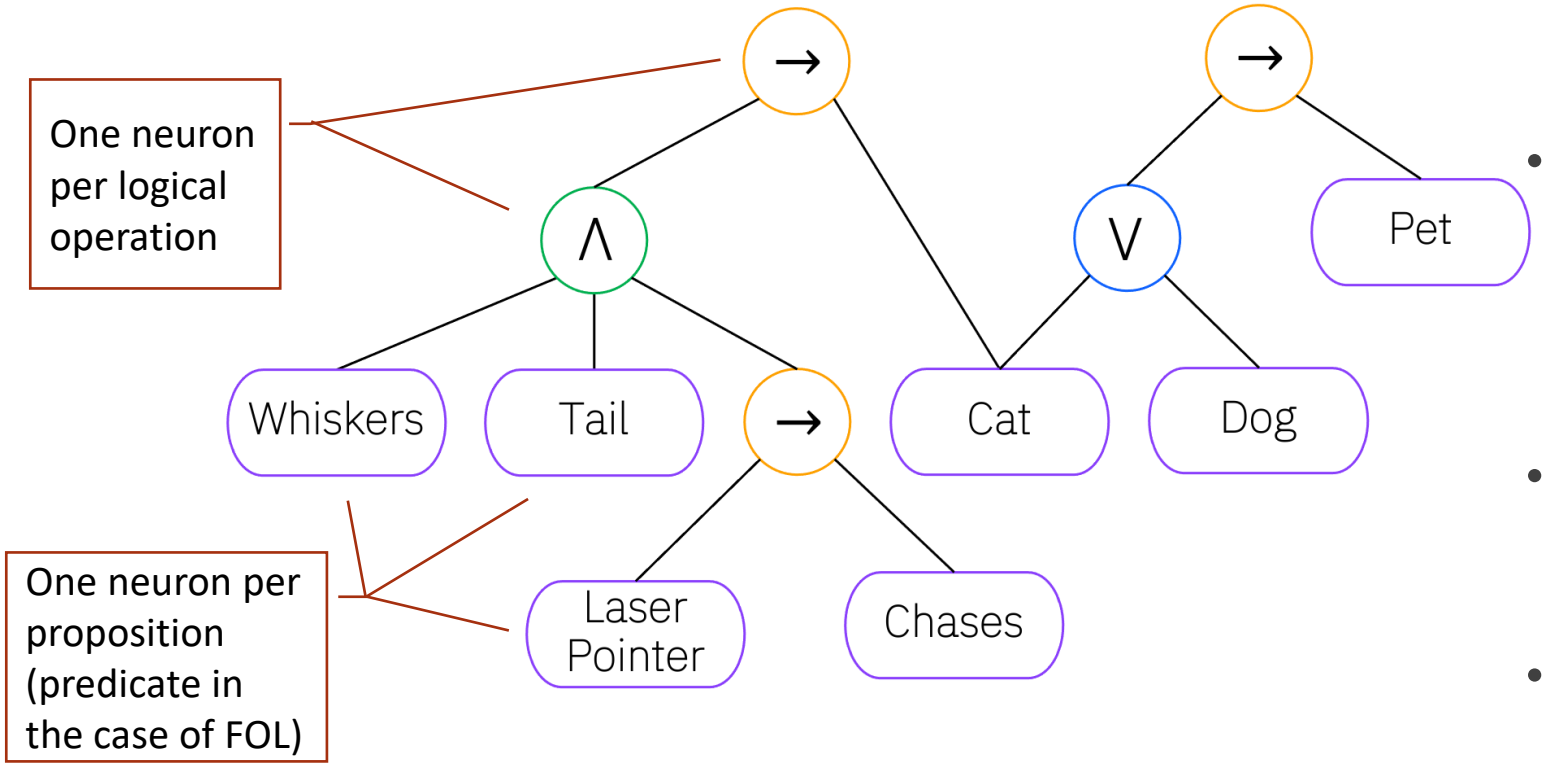
- $B \wedge C \wedge \neg D \rightarrow A$
- $E \wedge F \rightarrow A$
- B

converges to M(P) = {B}

Hitzler, P., Holldobler, S., Seda, A. K., ***Logic Programs and Connectionist Networks,*** *Journal ofApplied Logic, 2 (3), 2004, 273-300*

Logical Neural Network

- LNN is a graph made directly of their syntax tree of the rules
  - Each node returns a pair of values in [0,1] representing a lower and upper bound to their truth value



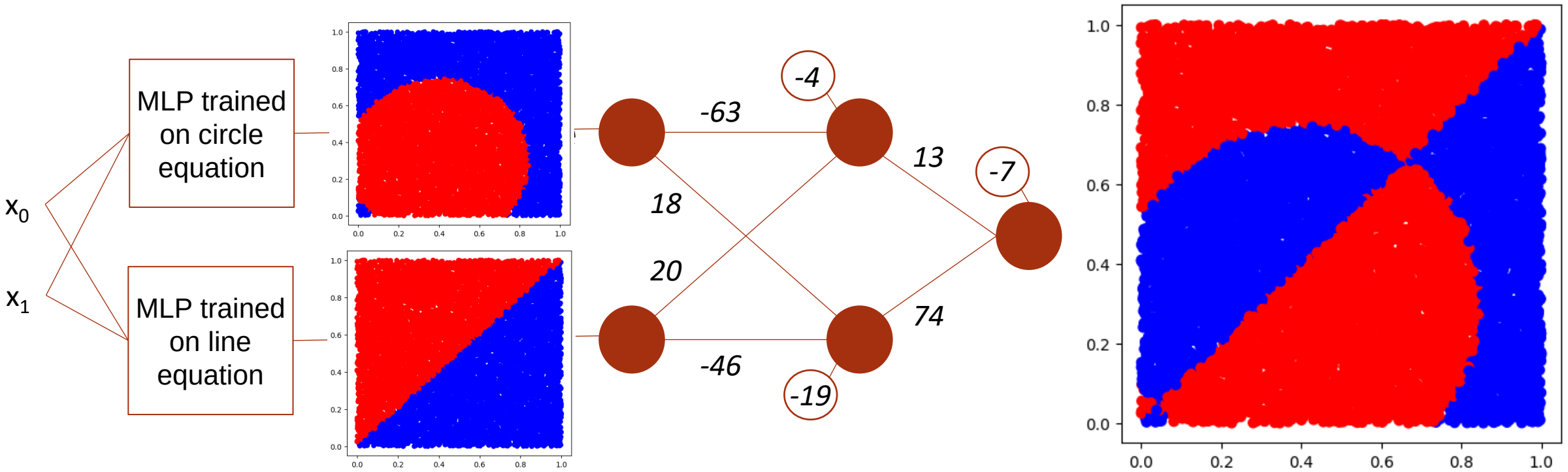
- Edges are weighted : importance-based weighted non-linear logic
- Connectives have bi-directional activation functions matching their respective truth functions with guaranties that the truth interval can only tighten
- Inference by alternating upward and downward passes until convergence
- FOL : each node keeps table of grounding tuples, each with bi-valued truth interval

R. Riegel, A. Gray, F. Luus, N. Khan, N. Makondo, I.Y. Akhalwaya, H. Qian, R. Fagin, F. Barahona, U. Sharma, S. Ikbal, H. Karanam, S. Neelam, A. Likhyani, S. Srivastava, ***Logical Neural Network*** *, arXiv:2006.13155.*

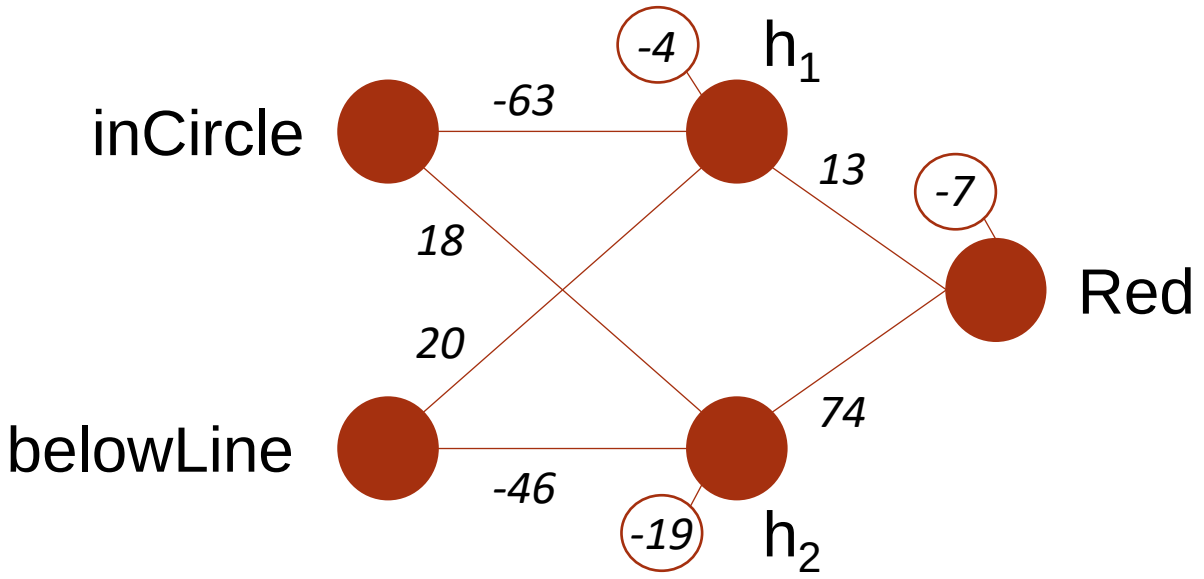
Take away (so far)

- If the rules and the concepts/predicates are known, a NN can be used for inference
  - Better than rule engines ?

Example



Example



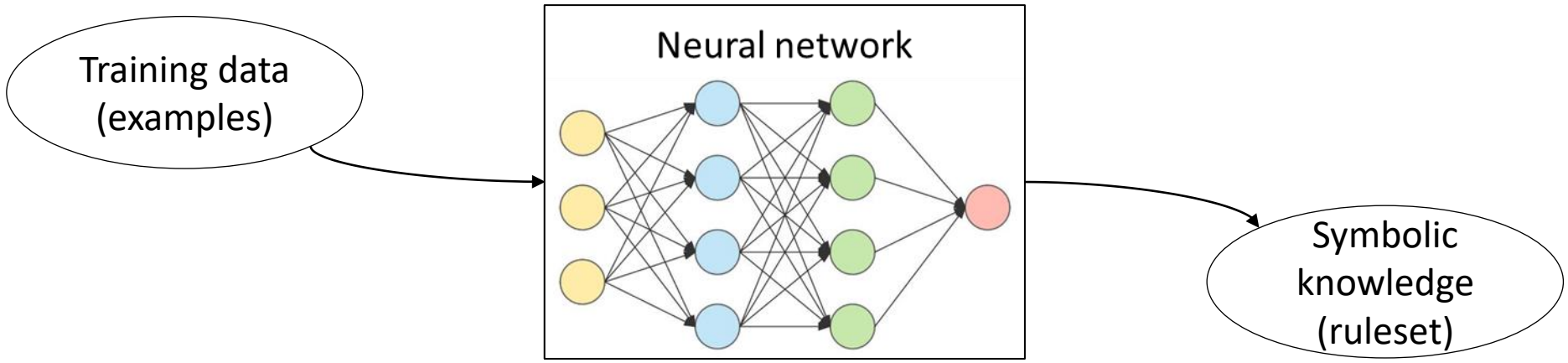
Ruleset

1. $h_1 \vee h_2 \rightarrow$ Red
2. $\neg$inCircle $\wedge$ belowLine $\rightarrow h_1$
3. $\neg$belowLine $\wedge$ inCircle $\rightarrow h_2$

If (point is in the circle and x is below the line)
or (point is outside of the circle and x is above the line)
then point is red
*(**else** point is blue)*

Rule extraction

- Training (learning rules)
- Decision explanation and documentation
- Validation (decision automation)

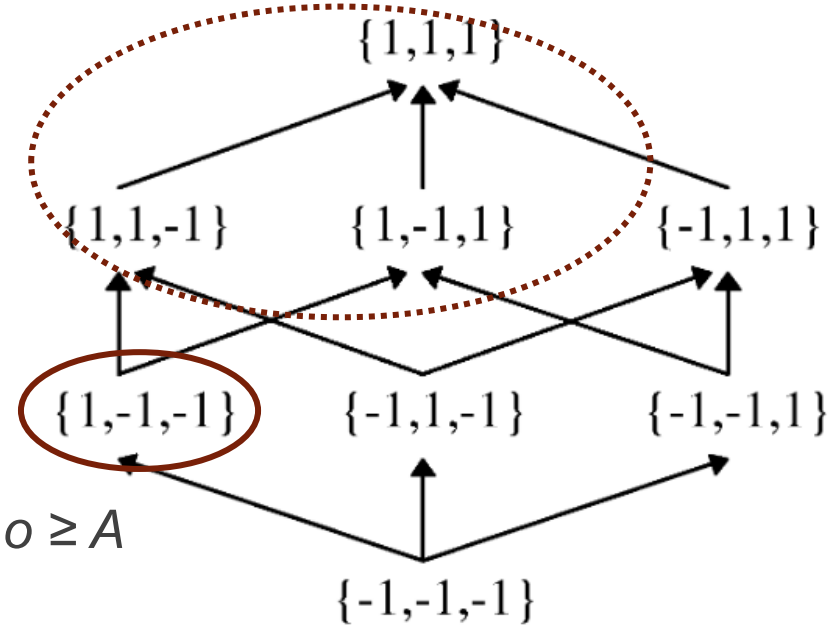


→ Decompositional VS pedagogical approaches ~ unpredictable accuracy VS exponential complexity

Rule extraction

Sketch of a pedagogical approach

- The output function *F(I)* induces a partial ordering on the input vectors *I = (i1… in)*, such that $I_1 \leq I_2$ iff $F(I_1) \leq F(I_2)$
- $I_1 \leq I_2 \Leftrightarrow \sum i_n \in I_1 \leq \sum i_n \in I_2$, if
  - *f* is monotonically increasing
  - all w^l_{ij} are positive
  - $i_n \in \{-1, 1\}$,
- $\bigwedge_{i_j=1, ij \in I} P(ij) \Rightarrow P(o)$ iff $F(I) \geq A$
  - under interpretation $P(i_n) = True$ iff $i_n = 1$ and $P(o) = True$ iff $o \geq A$
- Algorithm for positive network
  - Check if *P(o)* is always (resp. never) *True*, that is if $F(-1) \geq A$ (resp. $F(1) < A$)
  - Search the powerset $\mathcal{P}(\{i_1 \ldots i_n\})$ systematically, e.g. depth-first starting from $\varnothing$,
    - For $\gamma \in \mathcal{P}$, stop and move to next branch when $F(i_j = 1 \text{ if } j \in \gamma \text{ else } -1) \geq A$
    - Add the rule $\bigwedge_{ij \in \gamma} P(ij) \Rightarrow P(o)$ to the ruleset



A.S. d'Avila Garcez, K. Broda, D.M. Gabbay, ***Symbolic knowledge extraction from trained neural networks: A sound approach,*** *Artificial Intelligence 125 (2001) 155–207*

Rule extraction

Sketch of a pedagogical approach

- The algorithm is exact = sound and complete for positive and regular networks
  - Regular networks have only positive weights between the input and hidden layer, and only positive or only negative weight between the hidden and each ouput layer node
  - Networks with all positive or all negative weights from the hidden to the output layer can be regularized if no input node has both positive and negative weights to the hidden layer
    - If input node i_n has only negative weights to the hidden layer, flip them all to positive and flip the interpretation of $P(i_n)$ to $P(i_n) = True\ (False)$ iff $i_n = -1\ (1)$
- The algorithm can be generalized to non-regular networks with 1 hidden layer by considering each regular subnetwork separately and composing the results
  - Single perceptrons are trivially regular networks
- The generalized algorithm is sound, but incomplete

A.S. d'Avila Garcez, K. Broda, D.M. Gabbay, ***Symbolic knowledge extraction from trained neural networks: A sound approach,*** *Artificial Intelligence 125 (2001) 155–207*

Take away (so far)

- If the rules and the predicates are known, a NN can be used for inference
  - Better than rule engines ?
- If the predicates are known, a NN can be used to learn the rules
  - And the rules can be extracted from the NN (under conditions)
  - Better than symbolic rule learners ?

Example

Logic program

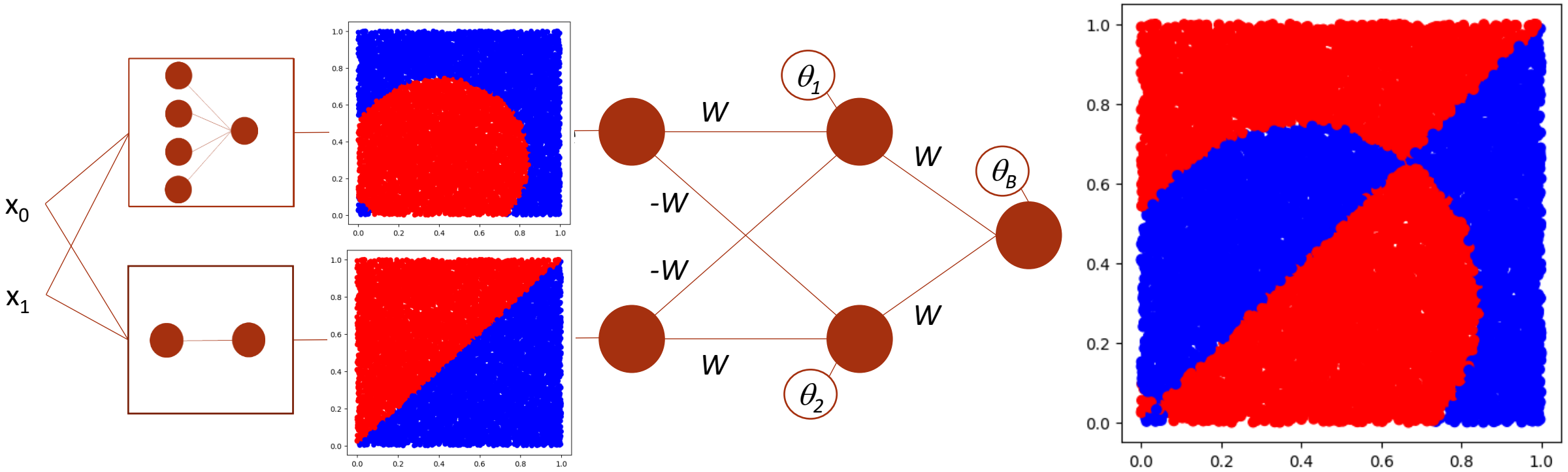
Red :- inCircle $\wedge$ $\neg$belowLine (1)

Red :- $\neg$inCircle $\wedge$ belowLine (2)

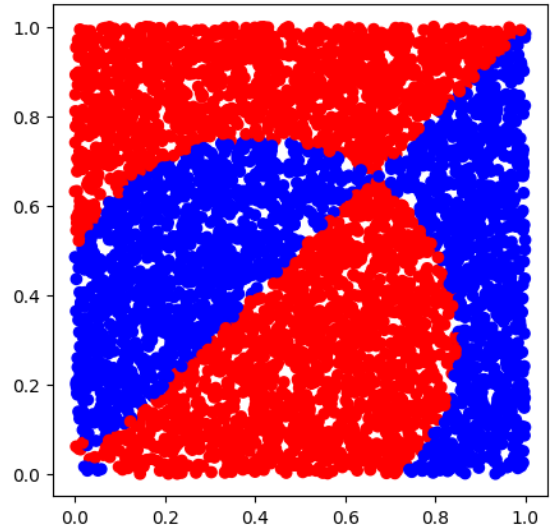
(Blue :- $\neg$Red

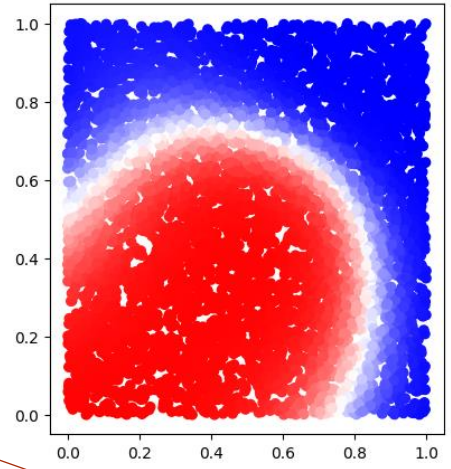
Parameters

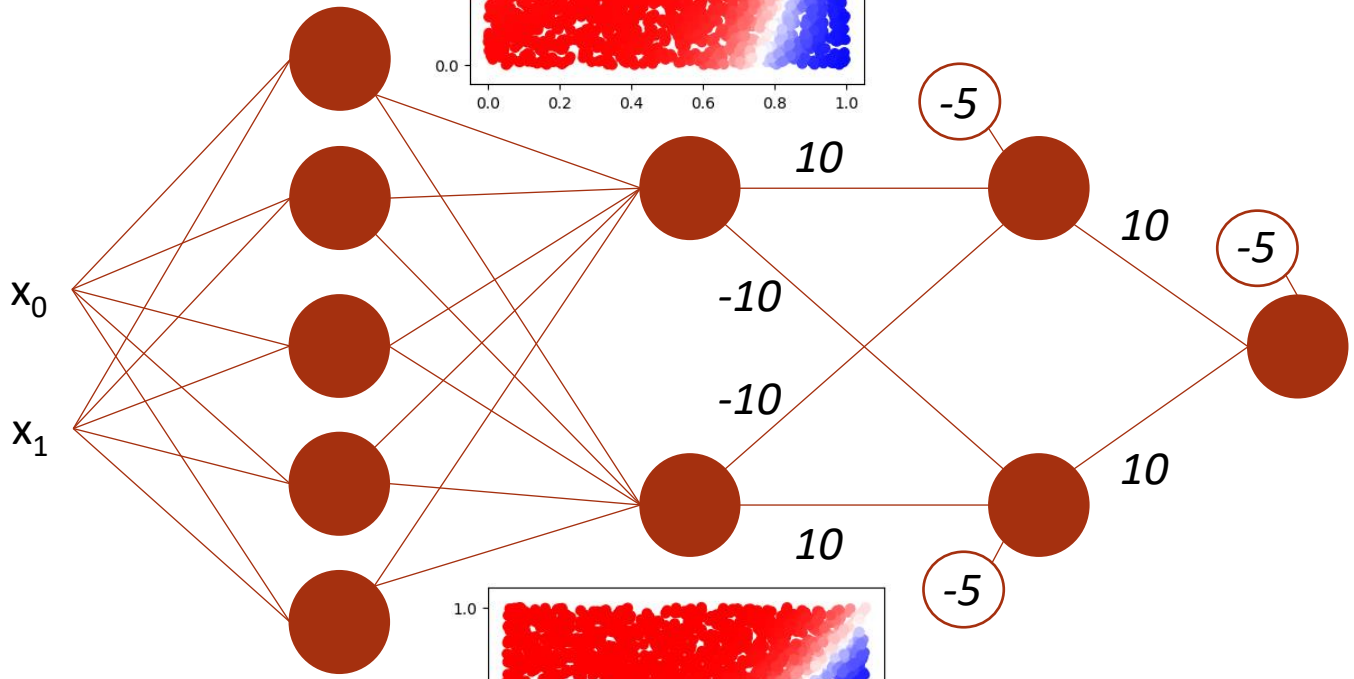
- W = 4.4
- $\theta_1 = \theta_2 = \theta_B = -3.3$

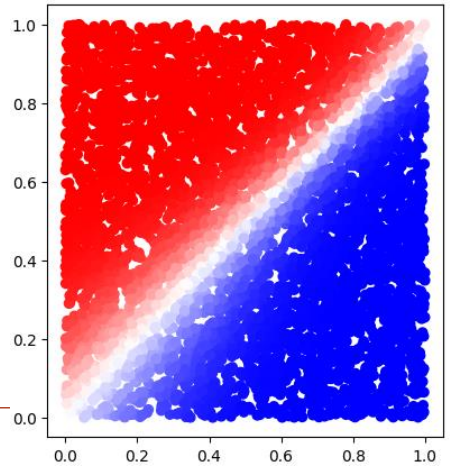


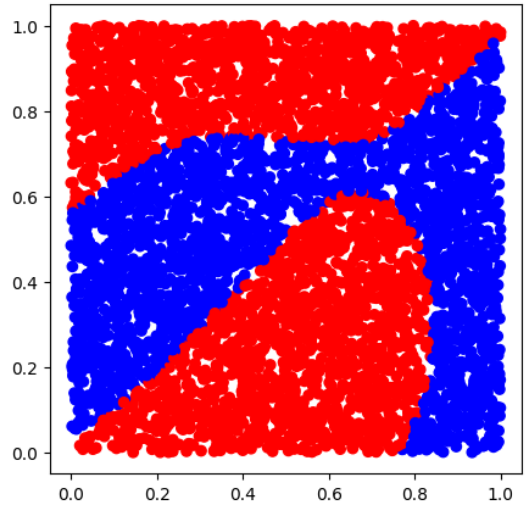
Example



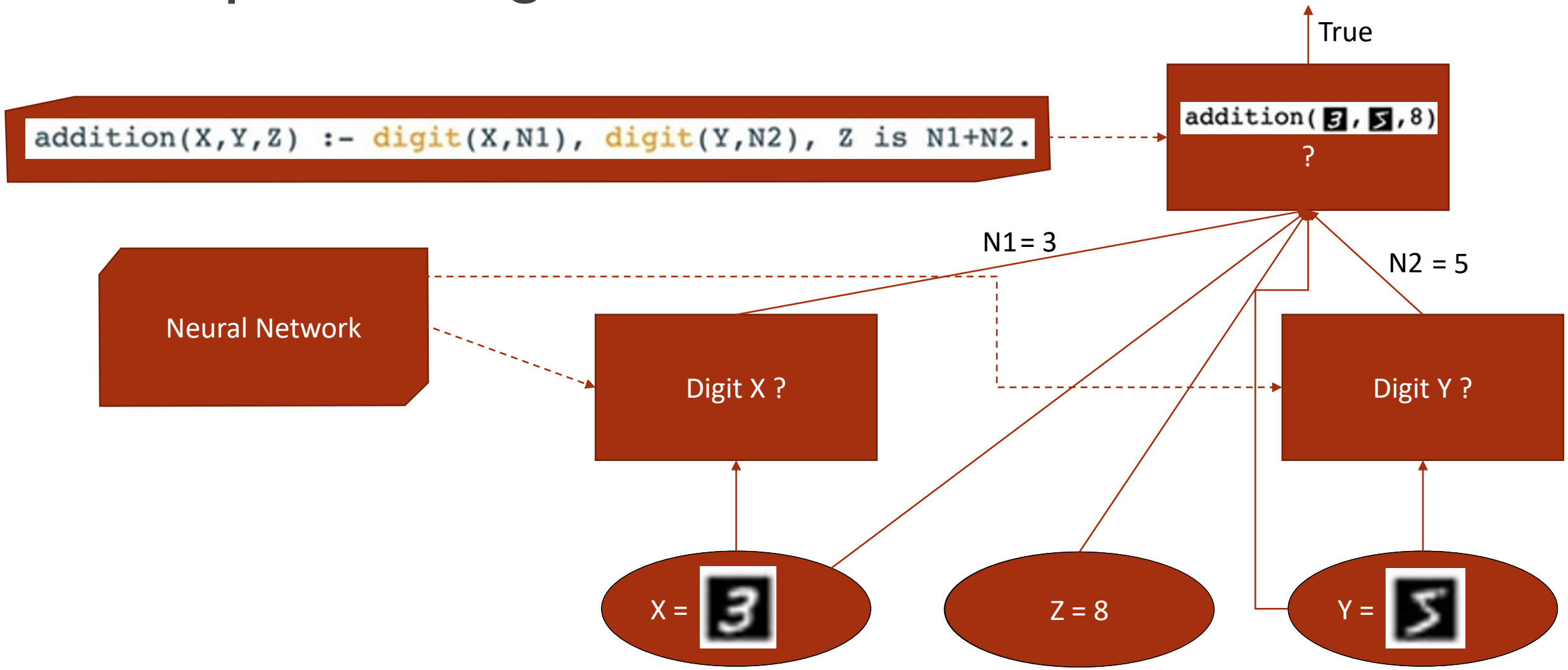








DeepProbLog (Deep Probabilistic Logic programming)



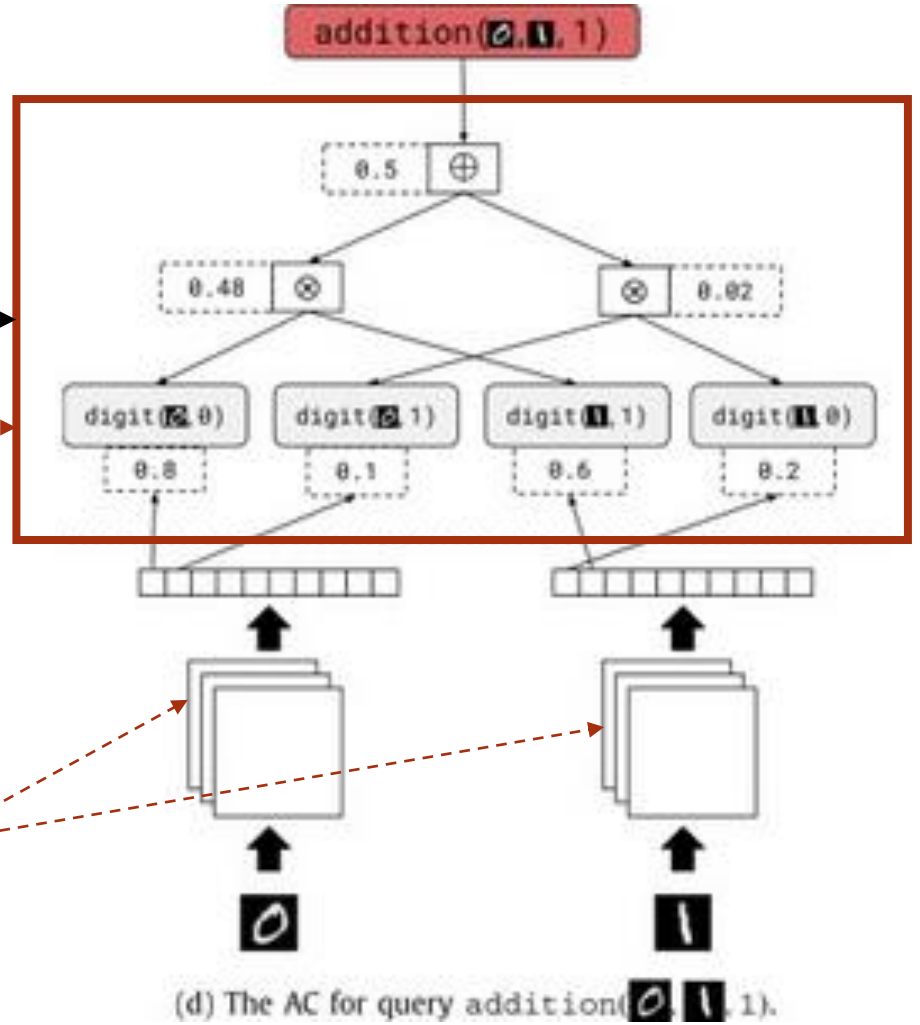
R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, L. De Raedt, ***DeepProbLog: Neural Probabilistic Logic Programming****, NeurIPS 2018*

DeepProbLog (Deep Probabilistic Logic programming)

Differentiable probabilistic logic programming engine

implements

the symbolic **Logic Program**

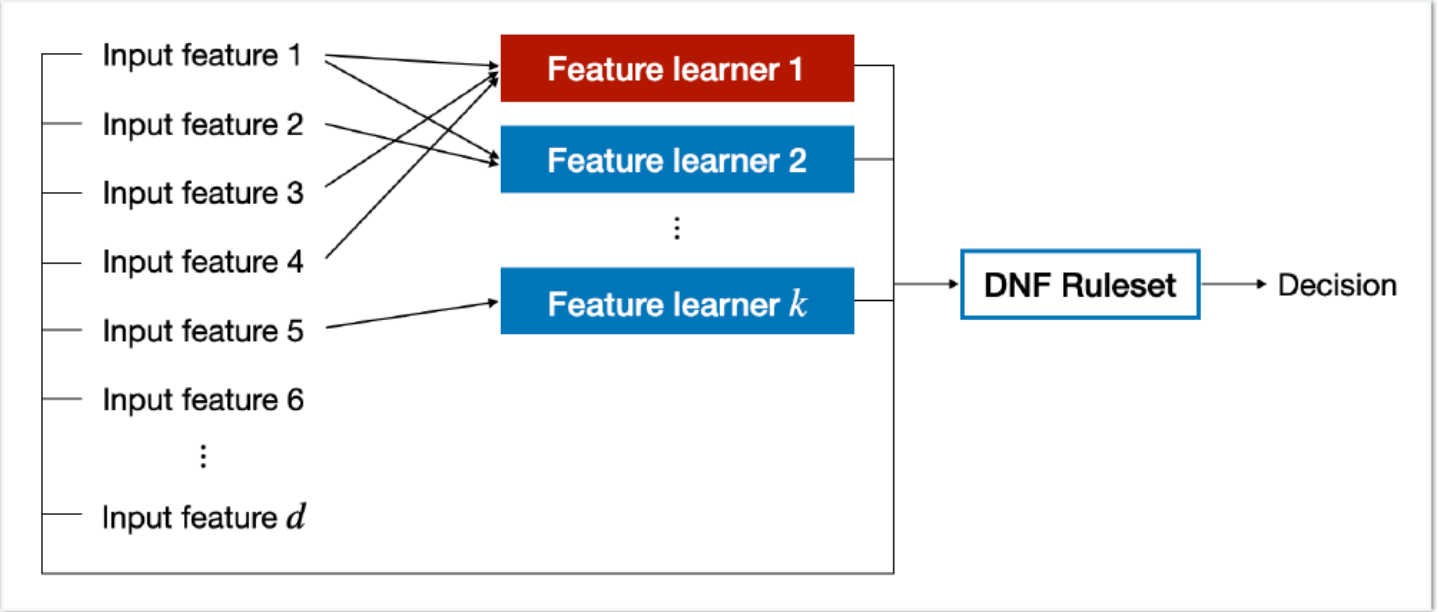


- Differentiability enables the back-propagation of the gradient of the loss
- And thus training of the **neural network** without explicit labelling of the images

R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, L. De Raedt, ***DeepProbLog: Neural Probabilistic Logic Programming****, NeurIPS 2018*

Extending to non-differentiable ruleset

- d input features: $x_1 \ldots x_d$
- k feature learners: $f_1 \ldots f_k$ where $f_i : X_i \rightarrow IR$ and X_i is a subset of the feature space. The learned features are denoted by $z_1 \ldots z_k$
- A fixed DNF ruleset that uses $x_1 \ldots x_d$ and $z_1 \ldots z_k$



- **Goal**: Updated feature model to improve the accuracy without modifying anything else
- **Problems to solve**: target value, attribution

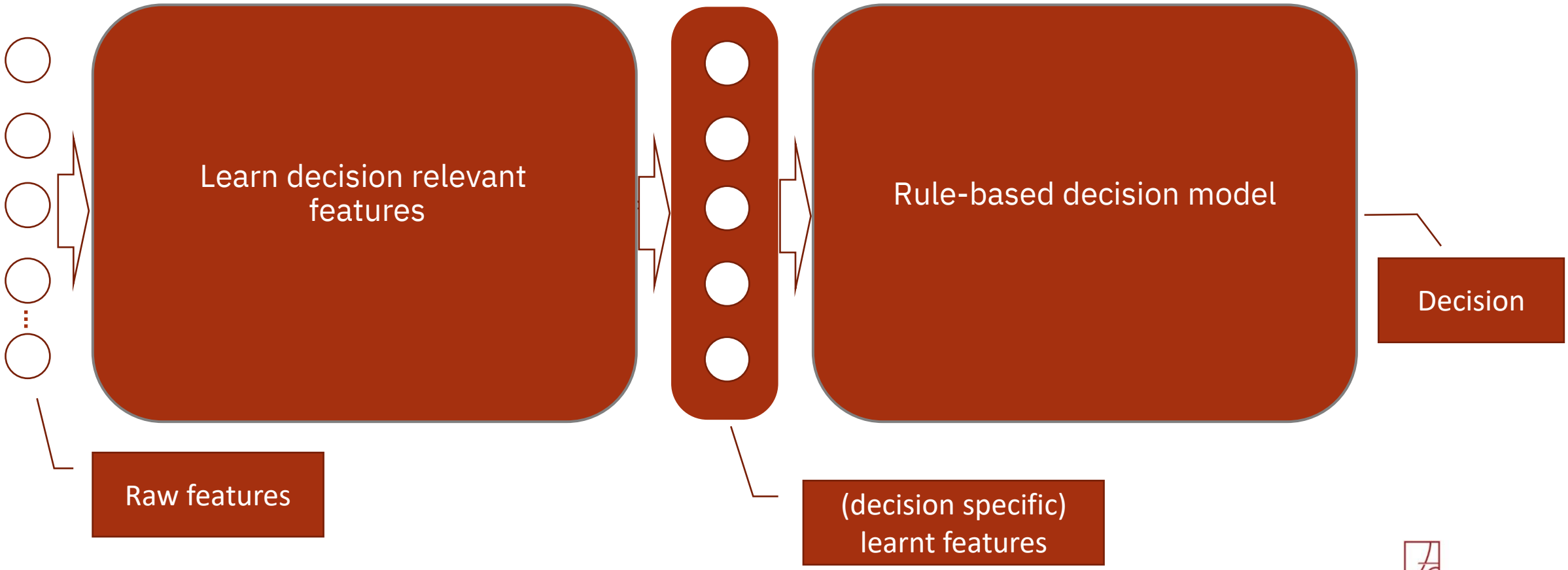
Take away (so far)

- If the rules and the predicates are known, a NN can be used for inference
  - Better than rule engine ?
- If the predicates are known, a NN can be used to learn the rules
  - And the rules can be extracted from the NN (under conditions)
  - Better than symbolic rule learners ?
- If the rules are known, NNs can be used to ground the predicates
  - At least if the inference engine is differentiable
  - And if a symbolic formulation of the grounding can be extracted*…

* *Ahmed M. Alaa and Mihaela van der Schaar,* ***Demystifying Black-box Models with Symbolic Metamodels,*** *NeurIPS 2019*

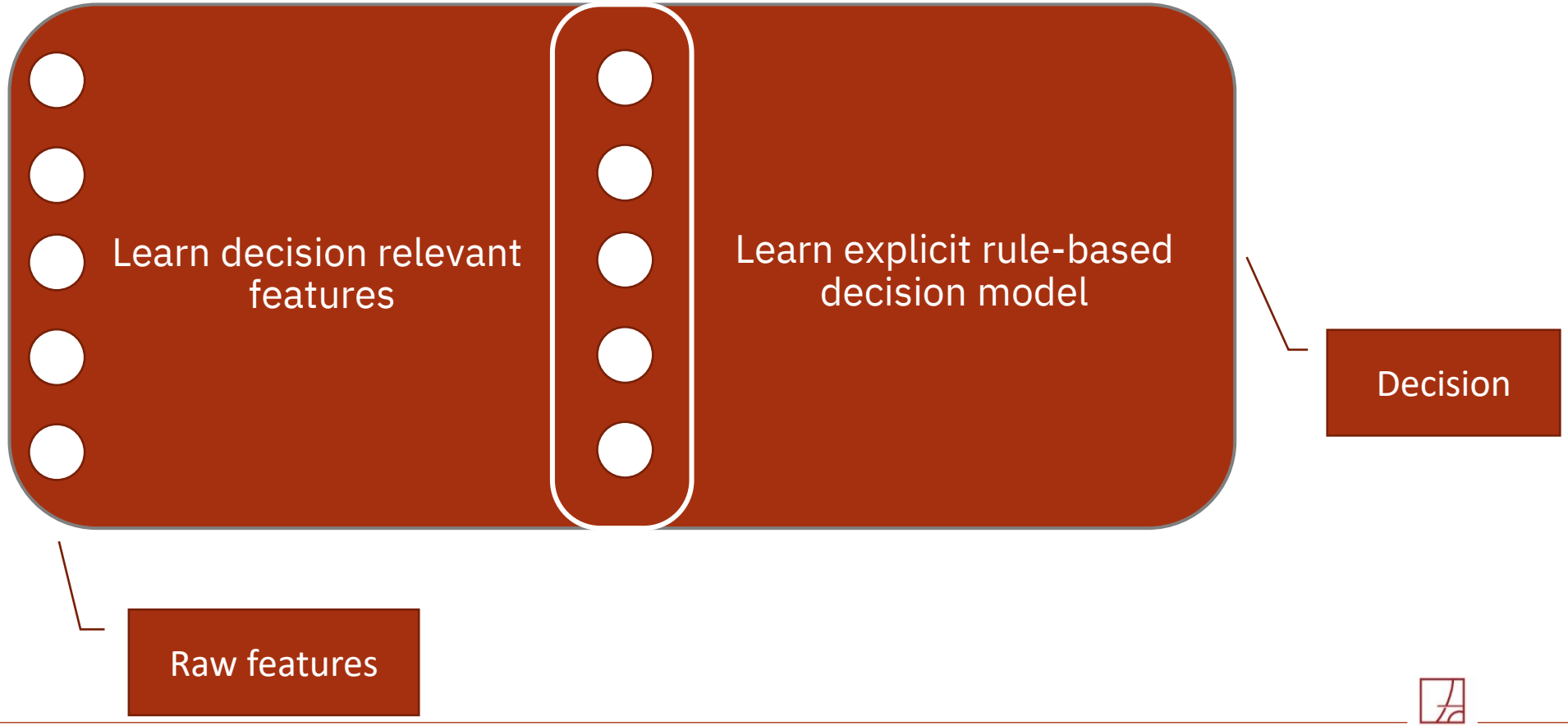
What if rules nor predicates are known ?

- We need to learn the vocabulary that is required to learn “good” rulesets for the decision at hand



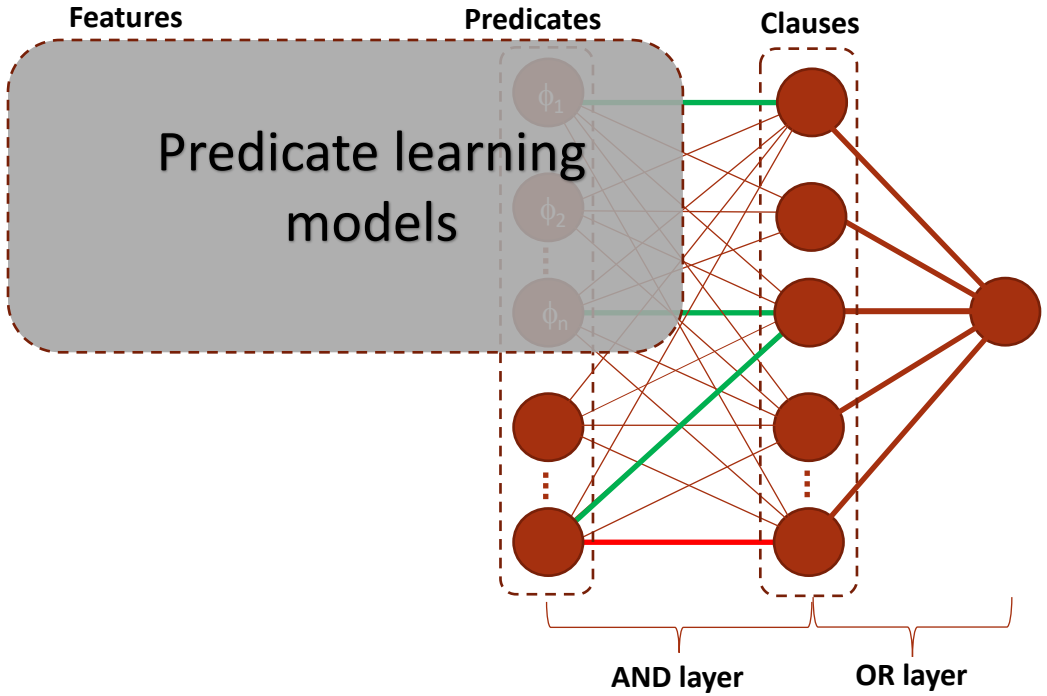
What if rules nor predicates are known ?

- We need to learn the vocabulary that is required to learn “good” rulesets for the decision at hand



Relational Rule Network (R2N)

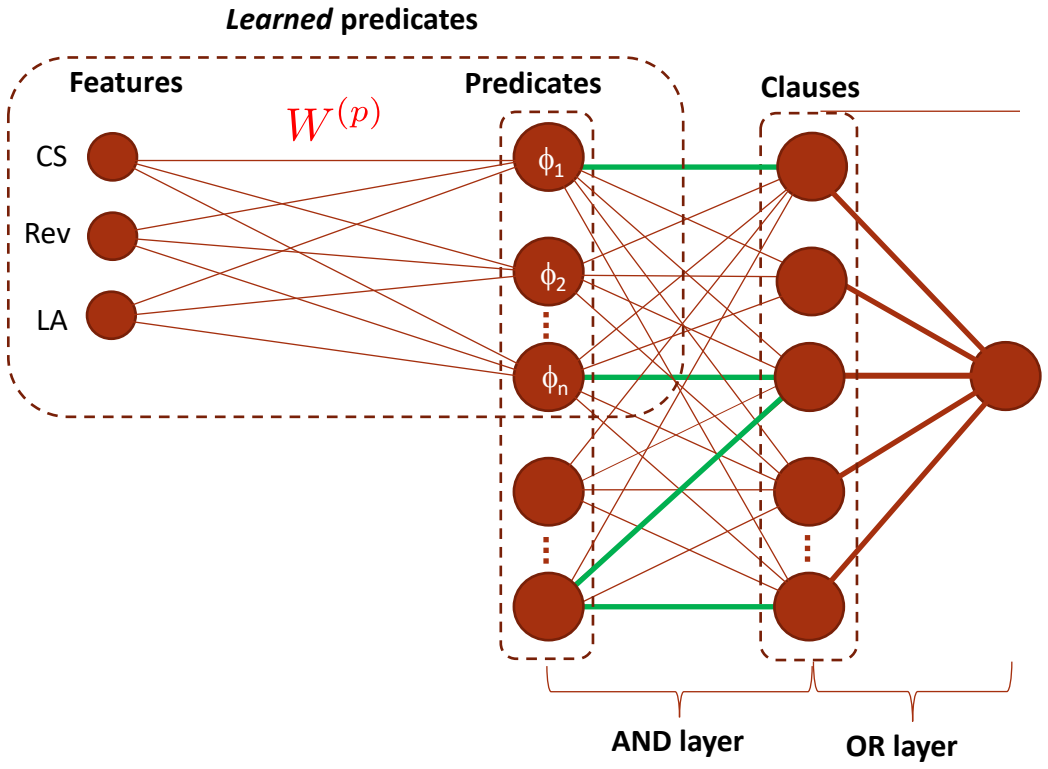
- Fully differentiable and trainable NN-architectures that maps one-to-one to a rule-based decision model “by design”
- Improve the rule language to enhance interpretability of the rules by learning the vocabulary used in these rule models



Remy Kusters, Yusik Kim, Marine Collery, Christian de Sainte Marie, Shubham Gupta, ***Differentiable Rule Induction with Learned Relational Features****, NeSy 22 – 16th International Workshop on Neural-Symbolic Learning and Reasoning, CEUR volume 3212, 30-44*

Relational Rule Network (R2N)

- Improve the rule language to enhance interpretability of the rules



Augment vocabulary with e.g. **linear combination of input features**

Univariate value comparisons, e.g.

$$[\mathrm{CS} > 100, \mathrm{Rev} > 25\mathrm{k}, ...]$$

↓

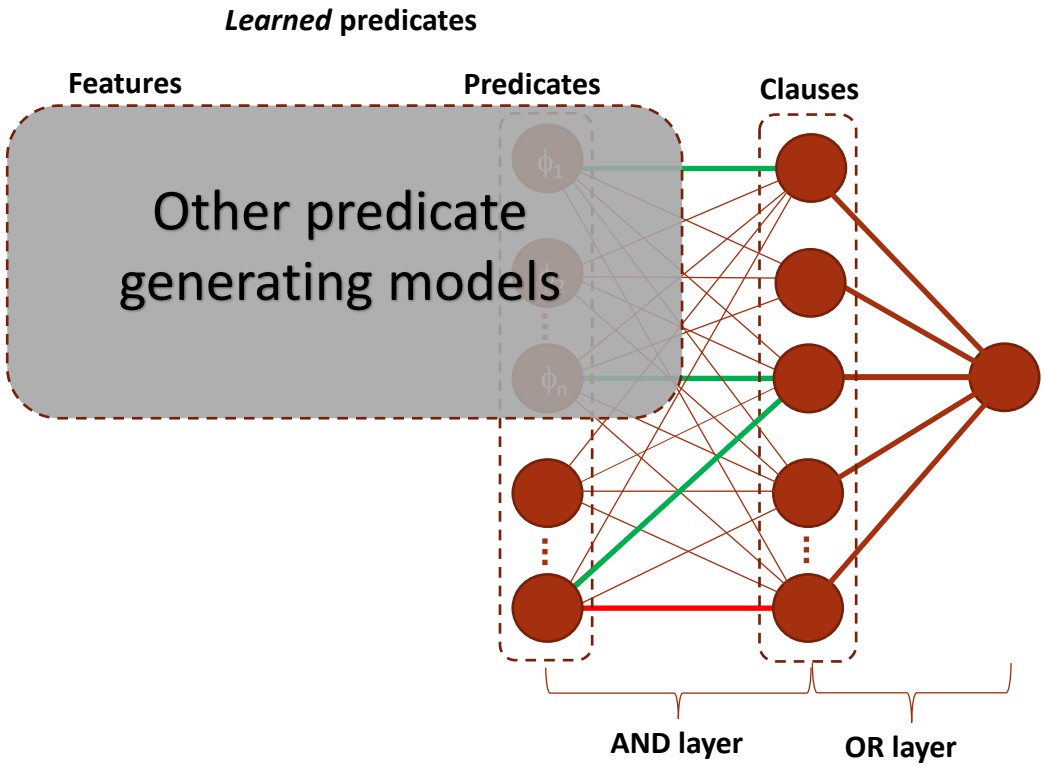
Multivariate value comparisons, e.g.

$$[w_0\mathrm{CS} + w_1\mathrm{Rev} + w_2\mathrm{LA} > 0, ...]$$

Remy Kusters, Yusik Kim, Marine Collery, Christian de Sainte Marie, Shubham Gupta, ***Differentiable Rule Induction with Learned Relational Features****, NeSy 22 – 16th International Workshop on Neural-Symbolic Learning and Reasoning, CEUR volume 3212, 30-44*

Relational Rule Network (R2N)

- Improve the rule language to enhance interpretability of the rules



Learning rules from:

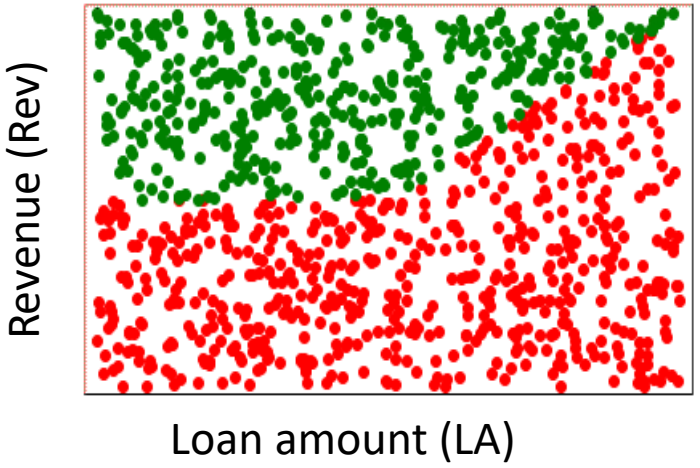
- **Aggregates**
- **Sequential data and time-dependent data**
- **Categorical data**

Adapting the **rule-language** to the problem at hand

Marine Collery, Remy Kusters, ***Learning binary classification rules for sequential data****, First International Workshop on Spatio-Temporal Reasoning and Learning, July 24, 2022, Vienna, Austria*

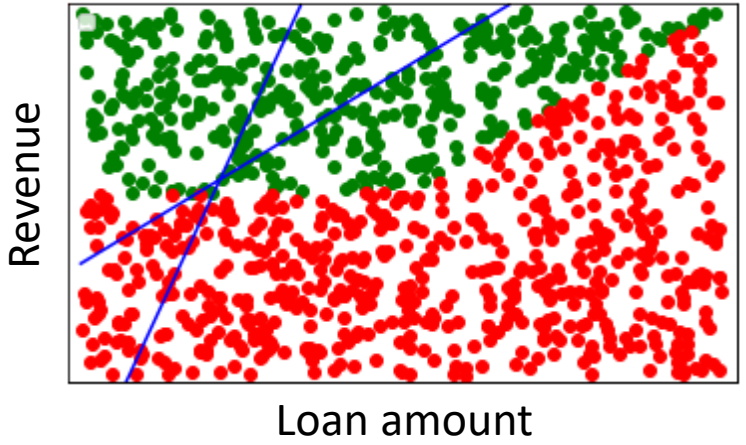
Relational Rule Network (R2N)

9 Clauses



IF Rev < 20k
OR (Rev < 50k **AND** LA > 20k)
OR (Rev < 25k **AND** LA > 10k)
OR (Rev < 75k **AND** LA > 35k)
OR …
THEN Label = 0 (Reject)

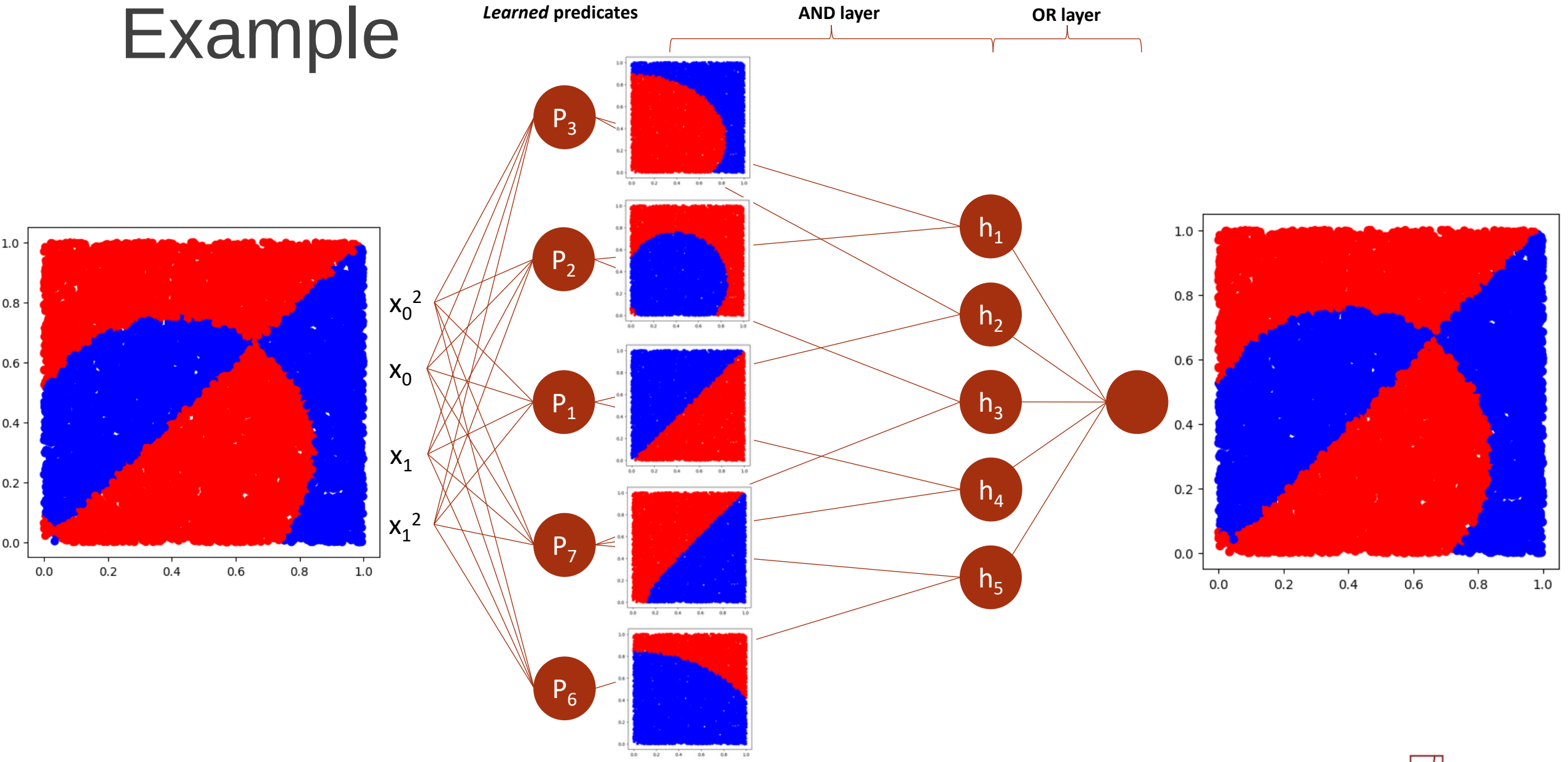
Multivariate predicates result in a more compact and interpretable representation



IF (Rev < 20k) **OR** (LA/Rev < 0.3)
THEN Label = 0 (Reject)

Remy Kusters, Yusik Kim, Marine Collery, Christian de Sainte Marie, Shubham Gupta, ***Differentiable Rule Induction with Learned Relational Features****, NeSy 22 – 16th International Workshop on Neural-Symbolic Learning and Reasoning, CEUR volume 3212, 30-44*

Example



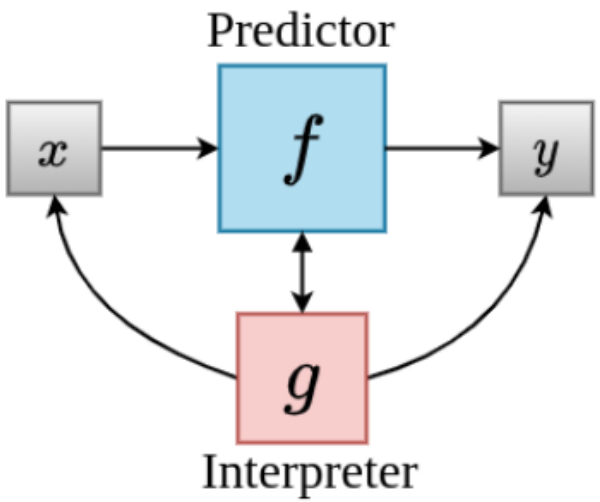
R2N open source code : ***https://github.com/IBM/symlearn***

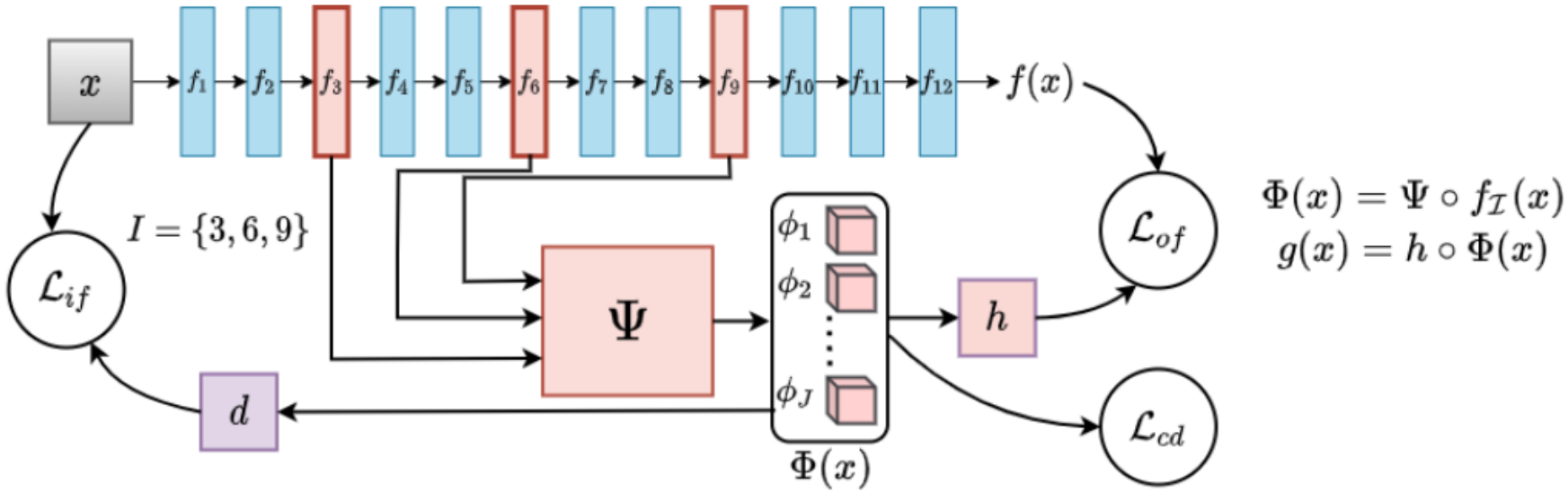
Take away (so far)

- If the rules and the predicates are known, a NN can be used for inference
  - Better than rule engine ?
- If the predicates are known, a NN can be used to learn the rules
  - And the rules can be extracted from the NN (under conditions)
  - Better than symbolic rule learners ?
- If the rules are known, NNs can be used to ground the predicates
  - At least if the inference engine is differentiable
  - And if a symbolic formulation of the grounding can be extracted…
- If neither the rules nor the predicates are known, specialized NNs can be used to learn the predicates
  - But…

We did find some rules and concepts

- But we did not find structure, e.g. conceptual models nor decision models
- In part because of lack of control on the level of abstraction of the representations that are learnt
- A layered architecture seems to be an interesting research direction





Jayneel Parekh 1, Pavlo Mozharovskyi 1, Florence d'Alché-Buc, ***A Framework to Learn with Interpretation****, NeurIPS 2021*

IBM Research

Thank you !
Questions ?

We are hiring !

→ contact me (csma@fr.ibm.com)

IBM