SKI: Symbolic Knowledge Injection

state of the art and research perspectives

Matteo Magnini
matteo.magnini@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum – Università di Bologna

07-06-2022

Next in Line...

Definition

We define symbolic knowledge injection as:

> *any* algorithmic *procedure affecting how sub-symbolic predictors draw their inferences in such a way that predictions are either* computed *as a function of, or made* consistent *with, some* given *symbolic knowledge*.*

* a wide definition that includes the vast majority of the works surveyed in [Besold et al., 2017, Xie et al., 2019, Calegari et al., 2020].

Symbolic Knowledge

A symbolic representation consists of: [van Gelder, 1990]

1. a set of symbols;
2. a set of grammatical rules governing the combining of symbols;
3. elementary symbols and any admissible combination of them can be assigned with meaning.

   $\Rightarrow$ Symbolic knowledge is both human and machine interpretable,

   - first order logic (FOL) is an example of symbolic representation.

Sub-symbolic data

- ML methods, and sub-symbolic approaches in general, represent data as arrays of real numbers, and knowledge as functions over such data;
- despite numbers are technically symbols as well, we cannot consider arrays and their functions as symbolic knowledge representation (KR) means;
- sub-symbolic approaches frequently violate Items 2 and 3.

Local vs distributed

When data are a numeric arrays:

Local representation

- Each number of the array has a well-defined meaning;
- example → iris dataset sample, array with 5 elements where each element has meaning (sepal/petal length/width and class).

Distributed representation

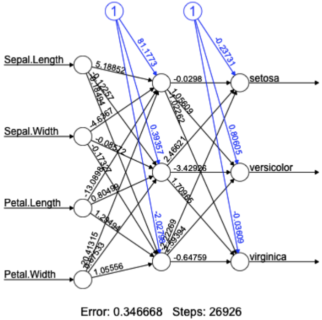
- Each number of the array is meaningless, unless it is considered along with its neighbourhood;
- example → images represented as $w \ x \ h$ matrices of numbers in range $[0, 1]$. (Violation of item 3)

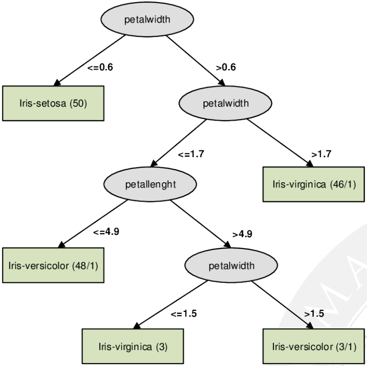
Sub-symbolic predictors I

- deep neural networks (DNN);
  - convolutional neural networks (CNN),
  - recurrent neural networks (RNN);
- kernel machines;
- basically everything that is sub-symbolic.

The vast majority of predictors are NN most probably because they are easy to manipulate and they have top performances.

Sub-symbolic predictors II





Why SKI?

There are several benefits:

- prevent the predictor to become a black-box!;
- reduce learning time;
- reduce the data size needed for training;
- improve predictor's accuracy;
- build a predictor that behave as a logic engine.

Explainable Artificial Intelligence [Gunning, 2016]

Explainability can be achieved:

Post-hoc explanation

- applying an algorithm of symbolic knowledge extraction on a trained predictor;
- output $\rightarrow$ logic rules that describe the predictor's behaviour.

By design

- constraining the behaviour of predictors that are natively black-boxes with symbolic knowledge;
- structuring the predictor's architecture with symbolic knowledge;
- output $\rightarrow$ a predictor that does not violate the prior knowledge.

Next in Line. . .

Aim

Enrich (learning support)

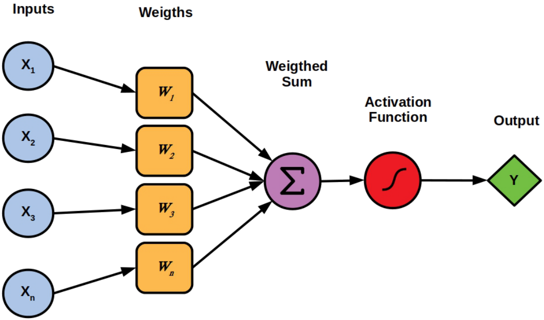
- reduce learning time;
- reduce the data size needed for training;
- improve predictor's accuracy.

Manifold (symbolic knowledge manipulation)

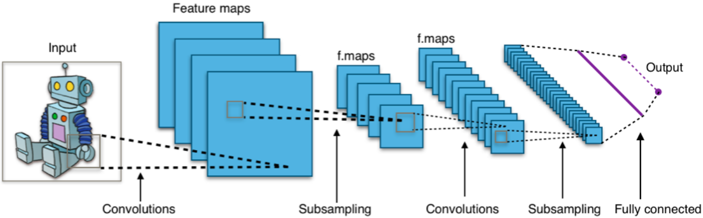
- logic inference;
- information retrieval;
- knowledge base completion/fusion.

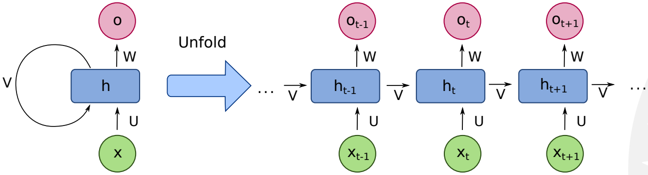
Predictors I

Theoretically, one can inject prior knowledge into any sub-symbolic predictor. In practice, NN are almost the sole predictors treated in literature, however, lot of different NN architecture are considered.



Predictors II





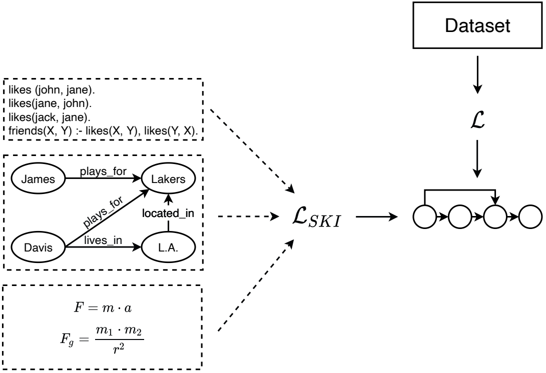
How

There exist three major ways to perform knowledge injection on sub-symbolic predictors:

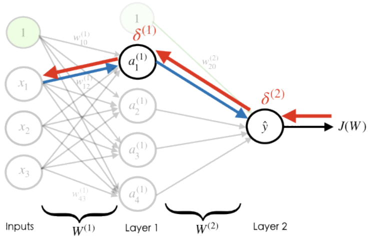
- constraining, a cost factor proportional to the violation of the knowledge is introduced during learning;
- structuring, the architecture of the predictor is built in such a way to mimic the knowledge;
- embedding, the symbolic knowledge is embedded into a tensor form and it is given in input as training data to the predictor.

Constraining I

- Knowledge cost factor is introduced in the loss function;
- for NN the cost affects backpropagation [Baldi and Sadowski, 2016] during training.
  - $\Rightarrow$ Predictor does not violate the prior knowledge (to a certain extent).



Constraining II



Forward pass
$$\begin{cases} z^{(1)} = W^{(1)}x + b_1 \\ a^{(1)} = \sigma(z^{(1)}) \\ z^{(2)} = W^{(2)}a^{(1)} + b_2 \\ a^{(2)} = \sigma(z^{(2)}) \\ \hat{y} = a^{(2)} \end{cases}$$

$$\frac{\partial J(W)}{\partial W_2} = \underbrace{\frac{\partial J(W)}{\partial a^{(2)}} * \frac{\partial a^{(2)}}{\partial z^{(2)}}}_{\delta^{(2)}} * \frac{\partial z^{(2)}}{\partial W_2}$$

$$\delta^{(2)} = a^{(2)} - y$$

$$\delta^{(1)} = \delta^{(2)}(W_2)^T \odot \sigma(z^{(1)})'$$

$$\frac{\partial J(W)}{\partial b_2} = \frac{\partial J(W)}{\partial a^{(2)}} * \frac{\partial a^{(2)}}{\partial z^{(2)}} * \frac{\partial z^{(2)}}{\partial b_2}$$

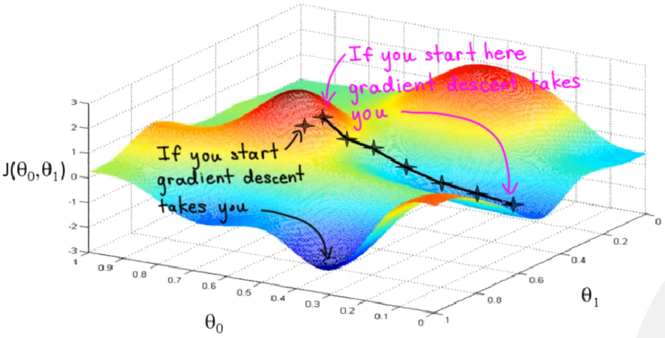
$$\frac{\partial J(W)}{\partial W_1} = \underbrace{\frac{\partial J(W)}{\partial a^{(2)}} * \frac{\partial a^{(2)}}{\partial z^{(2)}} * \frac{\partial z^{(2)}}{\partial a^{(1)}} * \frac{\partial a^{(1)}}{\partial z^{(1)}}}_{\delta^{(1)}} * \frac{\partial z^{(1)}}{\partial W_1}$$

$$\frac{\partial J(W)}{\partial b_1} = \frac{\partial J(W)}{\partial a^{(2)}} * \frac{\partial a^{(2)}}{\partial z^{(2)}} * \frac{\partial z^{(2)}}{\partial a^{(1)}} * \frac{\partial a^{(1)}}{\partial z^{(1)}} * \frac{\partial z^{(1)}}{\partial b_1}$$

$\longrightarrow$

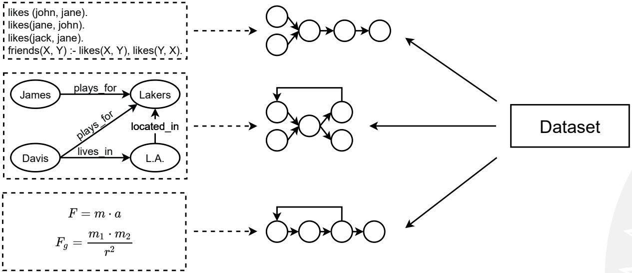
$$\boxed{\begin{aligned} \frac{\partial J(W)}{\partial W_2} &= \delta^{(2)} \odot a^{(1)} \\ \frac{\partial J(W)}{\partial b_2} &= \delta^{(2)} \\ \frac{\partial J(W)}{\partial W_1} &= \delta^{(1)} \odot x \\ \frac{\partial J(W)}{\partial b_1} &= \delta^{(1)} \end{aligned}}$$

Constraining III



Structuring I

- Inner architecture is shaped to be able to "mimic" the knowledge;
- for NN this means *ad-hoc* layers.
  - ⇒ Predictor directly exploits knowledge when needed.

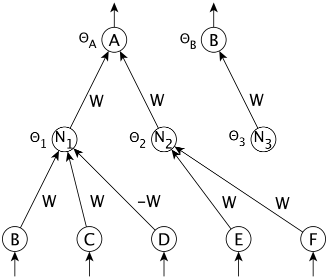


Structuring II

- We need to define a mapping from crispy logic rules into fuzzy continuous interpretations;
- then we need to map the interpretations into ad-hoc neurons/layers.

Structuring III

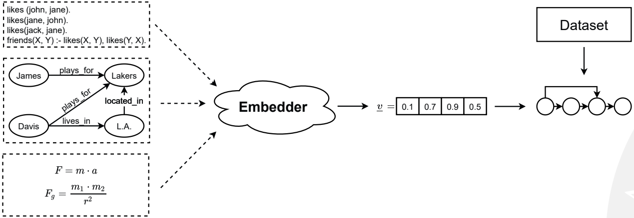
$$A \leftarrow B \land C \land \neg D.$$
$$A \leftarrow E \land F.$$
$$B \leftarrow \mathtt{true}.$$



Embedding I

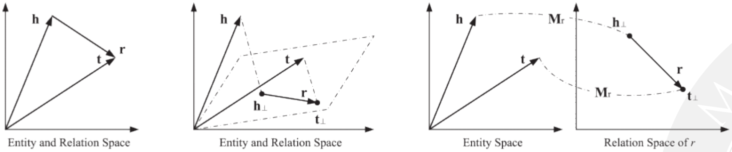
- Symbolic knowledge is embedded into a tensor form;
- this is used as predictor's input data (alone or with a "standard" dataset).

  $\Rightarrow$ Predictor's aim is manifold in most cases.



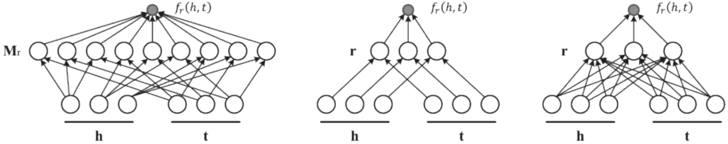
Embedding II

- Knowledge graph embedding [Wang et al., 2017];
- entities and relations are embedded into continuos vector spaces;
- scoring function $f_r(h, t)$ defined on each fact (h, r, t) to measure its plausibility;

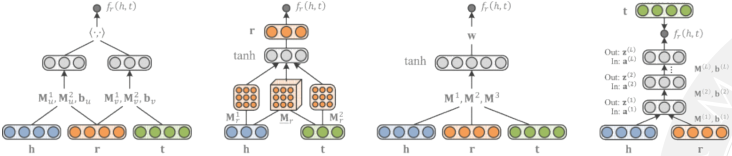


(a) TransE. (b) TransH. (c) TransR.

Embedding III



(a) RESCAL. (b) DistMult. (c) HolE.



(a) SME. (b) NTN. (c) MLP. (d) NAM.

Logic I

Intensional

- indirect representation of data,
- define a relation/set by describing its elements via other relations/sets.

Extensional

- direct representation of data,
- explicit definition of entities involved.

Recursive intensional predicates are very expressive and powerful, as they enable the description of infinite sets via a finite (and commonly small) amount of formulæ.

Logic II

Almost the totality of SKI algorithms deal with:

- first order logic (FOL);
- knowledge graph (KG);
- propositional logic (PL).

First Order Logic I

- FOL is extremely flexible and expressive;
- you can use recursion and define recursive structures;
- maybe too "powerful" for canonic NN.

  $\Rightarrow$ Most NN are natively DAG (directed acyclic graph)
  - this allows backpropagation as training algorithm but ...
  - how can you support recursion?

First Order Logic II

- FOL is extremely flexible and expressive;
- you can use recursion and define recursive structures;
- maybe too "powerful" for canonic NN.
  - $\Rightarrow$ Most NN are natively DAG (directed acyclic graph)
    - this allows backpropagation as training algorithm but ...
    - how can you support recursion?

You can't! Unless you use some tricks.

First Order Logic III

$$
\begin{aligned}
\mathit{parent}(\mathtt{abraham}, \mathtt{isaac}). \quad & \mathit{male}(\mathtt{abraham}). \\
\mathit{parent}(\mathtt{sarah}, \mathtt{isaac}). \quad & \mathit{female}(\mathtt{sarah}). \\
\mathit{parent}(\mathtt{isaac}, \mathtt{jacob}). \quad & \mathit{male}(\mathtt{isaac}). \\
\mathit{parent}(\mathtt{rebekah}, \mathtt{jacob}). \quad & \mathit{female}(\mathtt{rebekah}). \\
\ldots \quad & \mathit{male}(\mathtt{jacob}). \\
\forall X \forall Y\, \mathit{parent}(X, Y) \rightarrow & \; \mathit{child}(Y, X). \\
\forall X \forall Y\, \mathit{parent}(X, Y) \wedge \mathit{male}(X) \rightarrow & \; \mathit{father}(X, Y). \\
\forall X \forall Y\, \mathit{parent}(X, Y) \wedge \mathit{female}(X) \rightarrow & \; \mathit{mother}(X, Y). \\
\forall X \forall Y \exists Z\, \mathit{parent}(X, Z) \wedge \mathit{parent}(Z, Y) \rightarrow & \; \mathit{grandparent}(X, Y).
\end{aligned}
$$

Knowledge Graph I

- Only constants, variables and n-ary predicates with $n < 3$;
- collections of triplets $\langle \texttt{a}\ f\ \texttt{b} \rangle$ or $f(\texttt{a}, \texttt{b})$
- essentially directed graph:
  - nodes $\rightarrow$ individuals,
  - vertices $\rightarrow$ properties connecting individuals;
- may instantiate an ontology, i.e., a formal description of classes characterising a given domain.

Knowledge Graph II

⟨AlfredHitchcock, DirectorOf, Psycho⟩

Sir Alfred Joseph Hitchcock (13 August 1899 – 29 April 1980) was an English film director and producer, ...

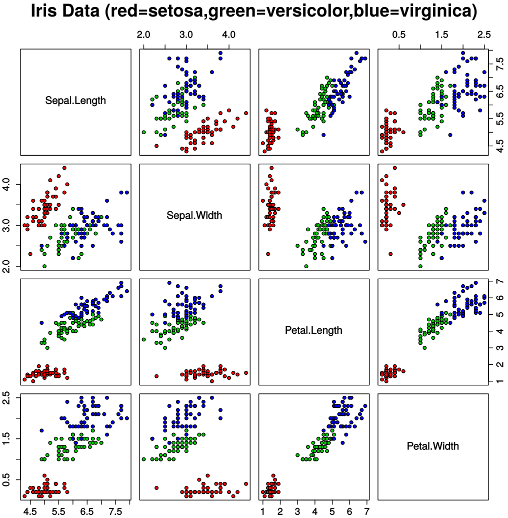
Psycho is a psychological horror film directed and produced by Alfred Hitchcock, and written by Joseph Stefano, ...

Propositional Logic I

- No quantifiers, terms, and non-atomic predicates;
- expressions involving one or many 0-ary predicates (propositions) possibly interconnected by ordinary logic connectives;
- low expressiveness, but easy to work with.

$$
\begin{aligned}
big\_petal \land average\_sepal &\rightarrow \texttt{virginica}. \\
big\_petal \land \neg average\_sepal &\rightarrow \texttt{versicolor}. \\
big\_petal &\rightarrow \texttt{setosa}. \\
average\_sepal &\equiv (3 \leq SepalWidth < 5) \\
big\_petal &\equiv (PetalLength > 3)
\end{aligned}
$$

Propositional Logic II



Next in Line. . .

Notable works

KBANN: Knowledge Base Artificial Neural Network [Towell and Shavlik, 1994]

It is one of the first works in SKI. Authors inject prior knowledge into a NN and validate their method on real world biological datasets.

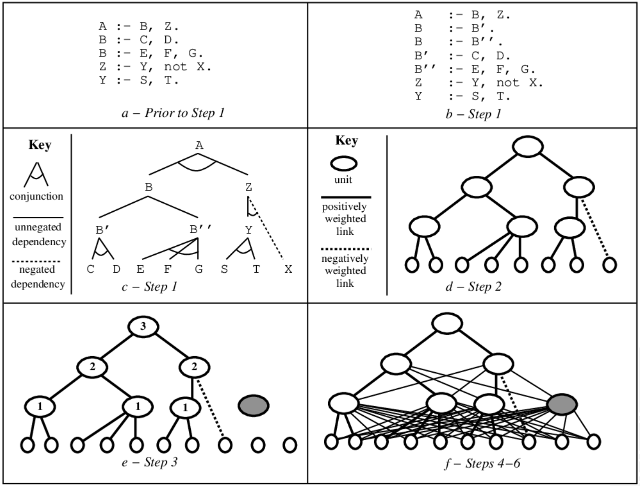
- aim → enrich;
- predictor → neural network;
- how → structuring and constraining;
- logic → propositional.

KBANN I

Algprithm

1. rewrite rules so that disjuncts are expressed as a set of rules that each have only one antecedent;
2. directly map the rule structure into a neural network;
3. label units in the KBANN-net according to their "level";
4. add hidden units to the network at user-specified levels (optional);
5. add units for known input features that are not referenced in the rules;
6. add links not specified by translation between all units in topologically-contiguous levels;
7. Perturb the network by adding near-zero random numbers to all link weights and biases.

KBANN II


A :- B, Z.
B :- C, D.
B :- E, F, G.
Z :- Y, not X.
Y :- S, T.
a – Prior to Step 1
A :- B, Z.
B :- B'.
B :- B''.
B' :- C, D.
B'' :- E, F, G.
Z :- Y, not X.
Y :- S, T.
b – Step 1
Key
conjunction
unnegated dependency
negated dependency
A
B
Z
B'
B''
Y
C
D
E
F
G
S
T
X
c – Step 1
Key
unit
positively weighted link
negatively weighted link
d – Step 2
3
2
2
1
1
1
e – Step 3
f – Steps 4–6

KBANN III

$$Error = -\sum_{i=1}^{n} \left[(1 - d_i) * \log_2 (1 - a_i) + d_i * \log_2 (a_i)\right]$$

$$Regularizer = \lambda \sum_{i \in \omega} \frac{(\omega_i - \omega_{init_i})^2}{1 + (\omega_i - \omega_{init_i})^2}$$

FANN I

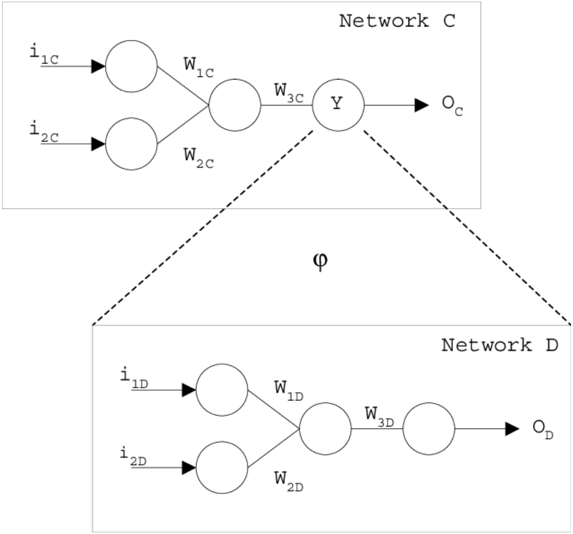
FANN: Fibred Artificial Neural Network

[d'Avila Garcez and Gabbay, 2004, Bader et al., 2005]

Authors present an interesting approach to deal with FOL in NN. The key idea is to allow single neurons to behave like entire embedded networks according to a fibring function ϕ.

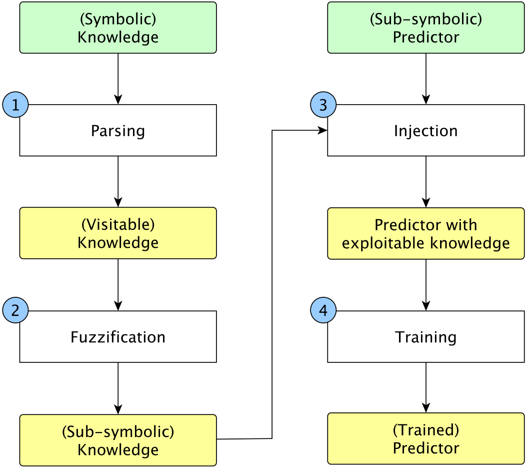
- aim $\rightarrow$ manifold;
- predictor $\rightarrow$ neural network;
- how $\rightarrow$ structuring;
- logic $\rightarrow$ first order logic.

FANN II


Network C
i1C
i2C
W1C
W2C
W3C
Y
OC
φ
Network D
i1D
i2D
W1D
W2D
W3D
OD

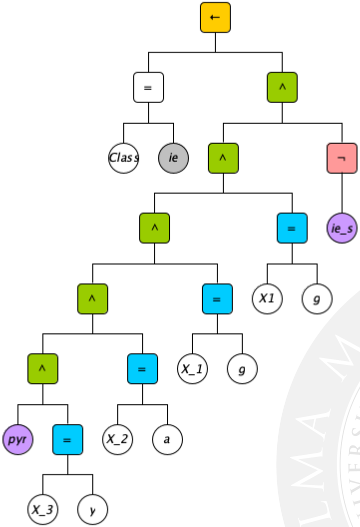
Next in Line...

General SKI workflow



1 – Parsing

$$
\begin{aligned}
& \mathit{class}(X_{-30}, \ldots, X_{30}, \mathit{ie}) \leftarrow \\
& \qquad \mathit{pyramidine\text{-}rich}(\ldots) \land \\
& \qquad X_{-3} = y \land \\
& \qquad X_{-2} = a \land \\
& \qquad X_{-1} = g \land \\
& \qquad X_{1} = g \land \\
& \qquad \neg(\mathit{ie\text{-}stop}(\ldots))
\end{aligned}
$$



2 – Fuzzification

Formula	Continuous interpretation
$[\![\neg\phi]\!]$	$1 - [\![\phi]\!]$
$[\![\phi \wedge \psi]\!]$	$min\{[\![\phi]\!], [\![\psi]\!]\}$
$[\![\phi \vee \psi]\!]$	$max\{[\![\phi]\!], [\![\psi]\!]\}$
$[\![\phi = \psi]\!]$	$[\![\neg(\phi \neq \psi)]\!]$
$[\![\phi \neq \psi]\!]$	$\lvert[\![\phi]\!] - [\![\psi]\!]\rvert$
$[\![\phi > \psi]\!]$	$max\{0, [\![\phi]\!] - [\![\psi]\!]\}$
$[\![\phi \geq \psi]\!]$	$[\![(\phi > \psi) \vee (\phi = \psi)]\!]$
$[\![\phi < \psi]\!]$	$max\{0, [\![\psi]\!] - [\![\phi]\!]\}$
$[\![\phi \leq \psi]\!]$	$[\![(\phi < \psi) \vee (\phi = \psi)]\!]$
$[\![\phi \Rightarrow \psi]\!]$	$min\{1, 1 - [\![\psi]\!] + [\![\phi]\!]\}$
$[\![\phi \Leftarrow \psi]\!]$	$min\{1, 1 - [\![\phi]\!] + [\![\psi]\!]\}$
$[\![\phi \Leftrightarrow \psi]\!]$	$min\{1, 1 - \lvert[\![\phi]\!] - [\![\psi]\!]\rvert\}$
$[\![\mathrm{expr}(\bar{X})]\!]$	$\mathrm{expr}([\![\bar{X}]\!])$
$[\![\mathtt{true}]\!]$	1
$[\![\mathtt{false}]\!]$	0
$[\![X]\!]$	x
$[\![\mathrm{k}]\!]$	k
$[\![p(\bar{X})]\!]^{**}$	$[\![\psi_1 \vee \ldots \vee \psi_k]\!]$
$[\![class(\bar{X}, \mathrm{y}_i) \leftarrow \psi]\!]$	$[\![\psi]\!]^*$

* encodes the value for the i^{th} output

** assuming p is defined by k clauses of the form: $p(\bar{X}) \leftarrow \psi_1, \ \ldots, \ p(\bar{X}) \leftarrow \psi_k$

$$class(X_{-30}, \ldots, X_{30}, ie) \leftarrow X_{-3} = y \wedge X_{-2} = a \wedge X_{-1} = g \wedge X_1 = g$$

$$\downarrow$$

$$min\{min\{min\{1 - |X_{-3} - y|, 1 - |X_{-2} - a|\}, 1 - |X_{-1} - g|\}, 1 - |X_1 - g|\}$$

3 – Injection

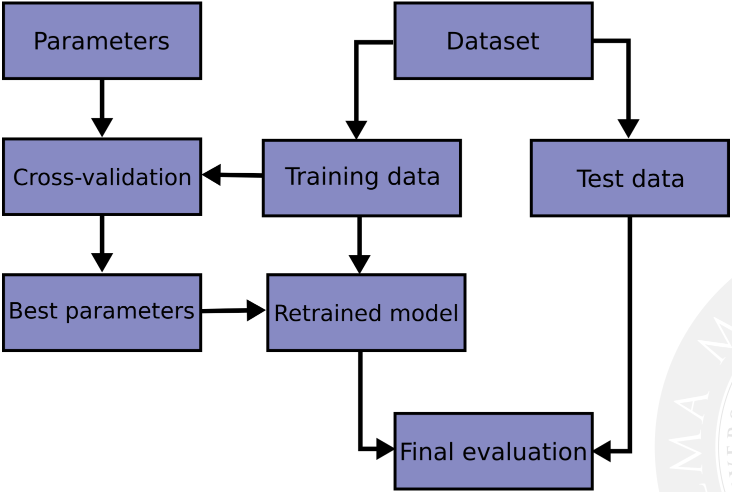
Injection step is algorithm specific but it falls back into the three approaches already discussed:

Injection families

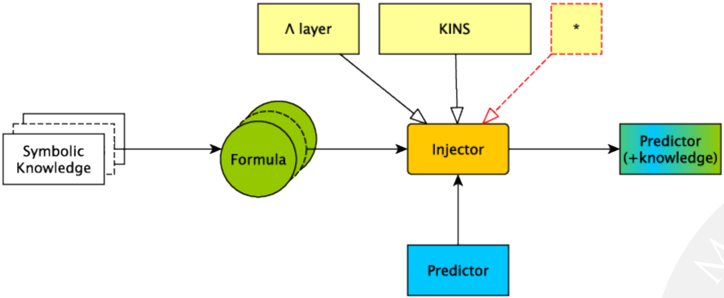
- constraining;
- structuring;
- embedding.

We will see some examples later.

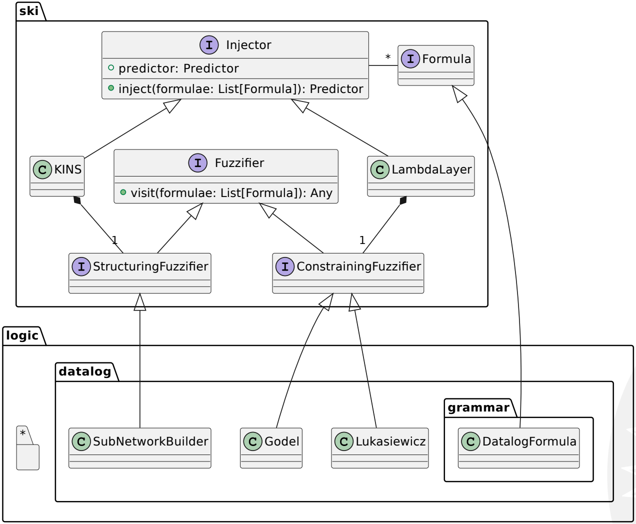
4 – Training


Parameters
Dataset
Cross-validation
Training data
Test data
Best parameters
Retrained model
Final evaluation

Overall Design I



Overall Design II



Overall Design III

Key components

- injector: an entity capable of injecting symbolic knowledge into sub-symbolic predictors;
- predictor: a classifier/regressor;
- formula: a visitable data structure representing a logic rule;
- fuzzifier: an entity that embed a crisp formula into a fuzzy continuous interpretation object.

Overall Design IV

```
from psyki.logic.datalog.grammar.adapters.antlr4 import get_formula_from_string
from psyki.ski.injectors import AnyInjector

# ...

# For this algorithm we need to explicitly specify the mapping
# between feature names and variable names
feature_mapping = {...}

# Symbolic knowledge
with open(filename) as f:
    rows = f.readlines()
# 1 - Parse textual logic rules into visitable Formulae
knowledge = [get_formula_from_string(row) for row in rows]

predictor = build_NN()
# 2 and 3 - Fuzzification and injection
injector = AnyInjector(predictor, feature_mapping, ...)
predictor_with_knowledge = injector.inject(knowledge)

# 4 - Training
predictor_with_knowledge.fit(train_x, train_y)
```

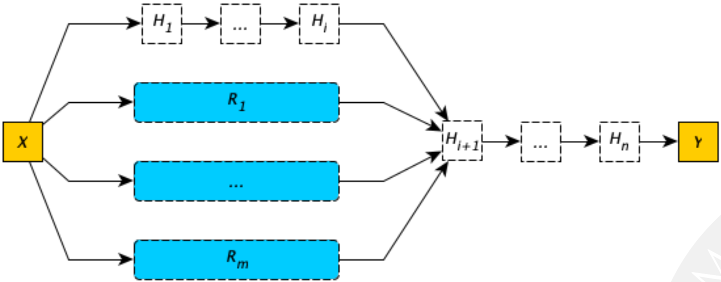
Knowledge Injection via Network Structuring I

KINS: Knowledge Injection via Network Structuring

A general SKI algorithm that does not impose constrains on the sub-symbolic predictor to enrich.

- aim $\rightarrow$ enrich;
- predictor $\rightarrow$ neural network;
- how $\rightarrow$ structuring;
- logic $\rightarrow$ stratified Datalog with negation.

Knowledge Injection via Network Structuring II



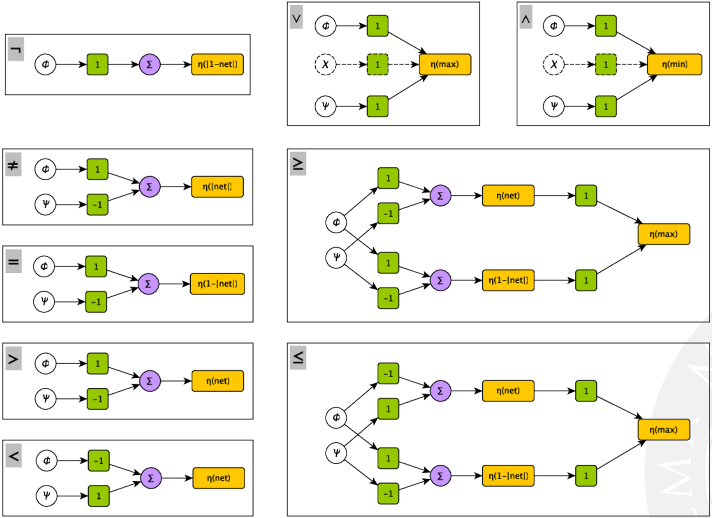
Knowledge Injection via Network Structuring III

Formula	C. interpretation	Formula	C. interpretation
$[\![\neg\phi]\!]$	$\eta\{1-[\![\phi]\!]\}$	$[\![\phi \leftarrow \psi]\!]$	$\eta\{min\{1, 1-[\![\phi]\!]+[\![\psi]\!]\}\}$
$[\![\phi \wedge \psi]\!]$	$\eta\{min\{[\![\phi]\!], [\![\psi]\!]\}\}$	$[\![\phi \leftrightarrow \psi]\!]$	$\eta\{min\{1, 1-\lvert[\![\phi]\!]-[\![\psi]\!]\rvert\}\}$
$[\![\phi \vee \psi]\!]$	$\eta\{max\{[\![\phi]\!], [\![\psi]\!]\}\}$	$[\![\mathrm{expr}(\bar{X})]\!]$	$\mathrm{expr}([\![\bar{X}]\!])$
$[\![\phi = \psi]\!]$	$\eta\{[\![\neg(\phi \neq \psi)]\!]\}$	$[\![\mathtt{true}]\!]$	1
$[\![\phi \neq \psi]\!]$	$\eta\{\lvert[\![\phi]\!]-[\![\psi]\!]\rvert\}$	$[\![\mathtt{false}]\!]$	0
$[\![\phi > \psi]\!]$	$\eta\{max\{0, [\![\phi]\!]-[\![\psi]\!]\}\}$	$[\![X]\!]$	x
$[\![\phi \geq \psi]\!]$	$\eta\{[\![(\phi > \psi) \vee (\phi = \psi)]\!]\}$	$[\![\mathtt{k}]\!]$	k
$[\![\phi < \psi]\!]$	$\eta\{max\{0, [\![\psi]\!]-[\![\phi]\!]\}\}$	$[\![p(\bar{X})]\!]^{**}$	$[\![\psi_1 \vee \ldots \vee \psi_k]\!]$
$[\![\phi \leq \psi]\!]$	$\eta\{[\![(\phi < \psi) \vee (\phi = \psi)]\!]\}$	$[\![class(\bar{X}, \mathtt{y}_i) \leftarrow \psi]\!]$	$[\![\psi]\!]^{*}$
$[\![\phi \rightarrow \psi]\!]$	$\eta\{min\{1, 1-[\![\psi]\!]+[\![\phi]\!]\}\}$		

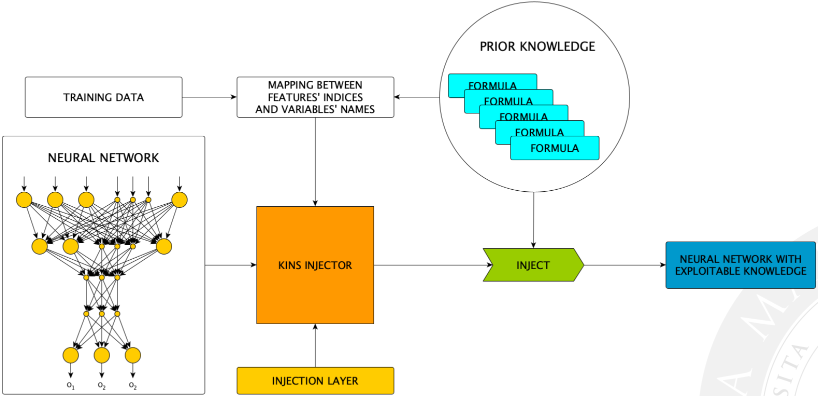
* encodes the value for the i^{th} output

** assuming p is defined by k clauses of the form:
$p(\bar{X}) \leftarrow \psi_1, \ \ldots, \ p(\bar{X}) \leftarrow \psi_k$

Knowledge Injection via Network Structuring IV



Knowledge Injection via Network Structuring V



Case study I

PSJGS: Primate Splice-Junction Gene Sequences dataset

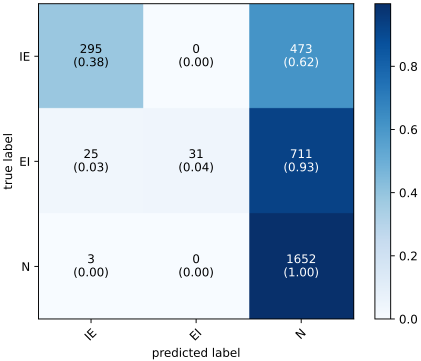
```
EI-stop ::- @-3 'TAA'.
EI-stop ::- @-3 'TAG'.
EI-stop ::- @-3 'TGA'.
EI-stop ::- @-4 'TAA'.
EI-stop ::- @-4 'TAG'.
EI-stop ::- @-4 'TGA'.
EI-stop ::- @-5 'TAA'.
EI-stop ::- @-5 'TAG'.
EI-stop ::- @-5 'TGA'.

IE-stop ::- @1 'TAA'.
IE-stop ::- @1 'TAG'.
IE-stop ::- @1 'TGA'.
IE-stop ::- @2 'TAA'.
IE-stop ::- @2 'TAG'.
IE-stop ::- @2 'TGA'.
IE-stop ::- @3 'TAA'.
IE-stop ::- @3 'TAG'.
IE-stop ::- @3 'TGA'.

pyramidine-rich :- 6 of (@-15 'YYYYYYYYYY').

EI :- @-3 'MAGGTRAGT', not(EI-stop).

IE :- pyramidine-rich, @-3 'YAGG', not(IE-stop).
```

```
Class, Id, DNA-sequence

EI,ATRINS-DONOR-521,CCAGCTGCAT...AGCCAGTCTG
EI,ATRINS-DONOR-905,AGACCCGCCG...GTGCCCCCGC
EI,BABAPOE-DONOR-30,GAGGTGAAGG...CACGGGGATG
...
IE,ATRINS-ACCEPTOR-701,TTCAGCGGCC...GCCCTGTGGA
IE,ATRINS-ACCEPTOR-1678,GGACCTGCTC...GGGGGCTCTA
IE,BABAPOE-ACCEPTOR-801,GCGGTTGATT...AAGATGAAGG
...
N,AGMKPNRSB-NEG-1,CAAAAGAACA...CAAGGCTACA
N,AGMORS12A-NEG-181,AGGGAGGTGT...GGGCATGGGG
N,AGMORS9A-NEG-481,TGGTCAATTC...TCTTGCTCTG
...

3190 Records
```

Case study II

Class	Logic Formulation
EI	$class(\bar{X}, \mathtt{ei}) \leftarrow X_{-3} = \mathtt{m} \wedge X_{-2} = \mathtt{a} \wedge X_{-1} = \mathtt{g} \wedge X_{+1} = \mathtt{g} \wedge$ $X_{+2} = \mathtt{t} \wedge X_{+3} = \mathtt{a} = \mathtt{r} \wedge X_{+4} = \mathtt{a} \wedge$ $X_{+5} = \mathtt{g} \wedge X_{+6} = \mathtt{t} \wedge \neg(ei\_stop(\bar{X}))$
	$ei\_stop(\bar{X}) \leftarrow X_{-3} = \mathtt{t} \wedge X_{-2} = \mathtt{a} \wedge X_{-1} = \mathtt{a}$
	$ei\_stop(\bar{X}) \leftarrow X_{-3} = \mathtt{t} \wedge X_{-2} = \mathtt{a} \wedge X_{-1} = \mathtt{g}$
	$ei\_stop(\bar{X}) \leftarrow X_{-3} = \mathtt{t} \wedge X_{-2} = \mathtt{g} \wedge X_{-1} = \mathtt{a}$
	$ei\_stop(\bar{X}) \leftarrow X_{-4} = \mathtt{t} \wedge X_{-3} = \mathtt{a} \wedge X_{-2} = \mathtt{a}$
	$ei\_stop(\bar{X}) \leftarrow X_{-4} = \mathtt{t} \wedge X_{-3} = \mathtt{a} \wedge X_{-2} = \mathtt{g}$
	$ei\_stop(\bar{X}) \leftarrow X_{-4} = \mathtt{t} \wedge X_{-3} = \mathtt{g} \wedge X_{-2} = \mathtt{a}$
	$ei\_stop(\bar{X}) \leftarrow X_{-5} = \mathtt{t} \wedge X_{-4} = \mathtt{a} \wedge X_{-3} = \mathtt{a}$
	$ei\_stop(\bar{X}) \leftarrow X_{-5} = \mathtt{t} \wedge X_{-4} = \mathtt{a} \wedge X_{-3} = \mathtt{g}$
	$ei\_stop(\bar{X}) \leftarrow X_{-5} = \mathtt{t} \wedge X_{-4} = \mathtt{g} \wedge X_{-3} = \mathtt{a}$
IE	$class(\bar{X}, \mathtt{ie}) \leftarrow pyramidine\_rich(\bar{X}) \wedge \neg(ie\_stop(\bar{X})) \wedge$ $X_{-3} = \mathtt{y} \wedge X_{-2} = \mathtt{a} \wedge X_{-1} = \mathtt{g} \wedge X_{+1} = \mathtt{g}$
	$pyramidine\_rich(\bar{X}) \leftarrow 6 \leq (X_{-15} = \mathtt{y} + \ldots + X_{-6} = \mathtt{y})$
	$ie\_stop(\bar{X}) \leftarrow X_{+2} = \mathtt{t} \wedge X_{+3} = \mathtt{a} \wedge X_{+4} = \mathtt{a}$
	$ie\_stop(\bar{X}) \leftarrow X_{+2} = \mathtt{t} \wedge X_{+3} = \mathtt{a} \wedge X_{+4} = \mathtt{g}$
	$ie\_stop(\bar{X}) \leftarrow X_{+2} = \mathtt{t} \wedge X_{+3} = \mathtt{g} \wedge X_{+4} = \mathtt{a}$
	$ie\_stop(\bar{X}) \leftarrow X_{+3} = \mathtt{t} \wedge X_{+4} = \mathtt{a} \wedge X_{+5} = \mathtt{a}$
	$ie\_stop(\bar{X}) \leftarrow X_{+3} = \mathtt{t} \wedge X_{+4} = \mathtt{a} \wedge X_{+5} = \mathtt{g}$
	$ie\_stop(\bar{X}) \leftarrow X_{+3} = \mathtt{t} \wedge X_{+4} = \mathtt{g} \wedge X_{+5} = \mathtt{a}$
	$ie\_stop(\bar{X}) \leftarrow X_{+4} = \mathtt{t} \wedge X_{+5} = \mathtt{a} \wedge X_{+6} = \mathtt{a}$
	$ie\_stop(\bar{X}) \leftarrow X_{+4} = \mathtt{t} \wedge X_{+5} = \mathtt{a} \wedge X_{+6} = \mathtt{g}$
	$ie\_stop(\bar{X}) \leftarrow X_{+4} = \mathtt{t} \wedge X_{+5} = \mathtt{g} \wedge X_{+6} = \mathtt{a}$

Case study III



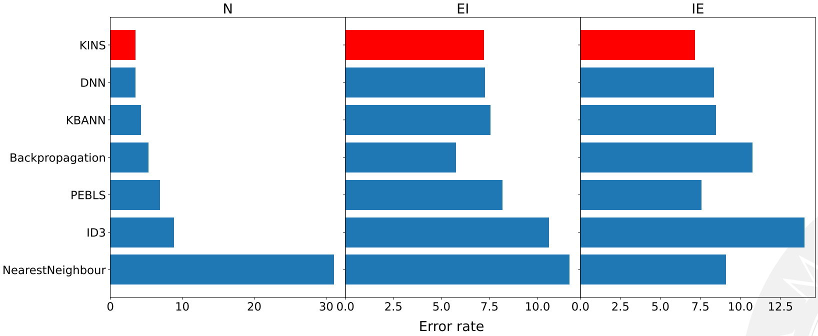
Case study IV

```
set_seed(self.seed)
# Loading dataset and apply one-hot encoding for each feature
# This means that for feature i_th we have 4 new features,
# one for each base: i_th_a, i_th_c, i_th_g, i_th_t.
data = get_splice_junction_data('data')
y = data_to_int(data.iloc[:, -1:], CLASS_MAPPING)
x = get_binary_data(data.iloc[:, :-1], AGGREGATE_FEATURE_MAPPING)
y.columns = [x.shape[1]]
data = x.join(y)
# Loading rules and conversion in Datalog form
rules = get_splice_junction_rules('kb')
rules = get_splice_junction_datalog_rules(rules)
rules = get_binary_datalog_rules(rules)
rules = [get_formula_from_string(rule) for rule in rules]
# Creation of the base model
model = create_fully_connected_nn_with_dropout()
injector = NetworkComposer(model, get_splice_junction_extended_feature_mapping()) # aka KINS!
result = k_fold_cross_validation(data, injector, rules, seed=self.seed)
result.to_csv(self.file + '.csv', sep=';')
```

Case study V

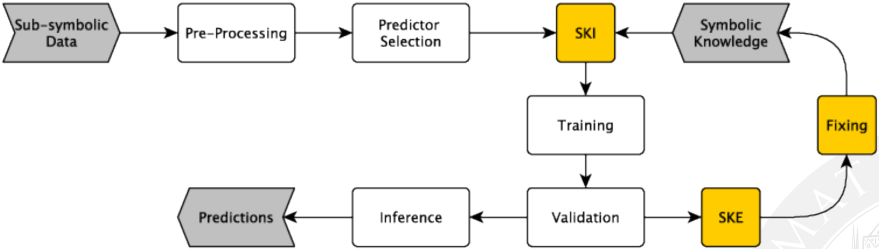
- neural network: 3-layers fully connected (64, 32, 3 neurons per layer respectively) with a 20% of dropout;
- mapping between features and variables: a map where keys are variables' names (e.g., $X_{-30}a, X_{-30}c, X_{-30}g, X_{-30}t, X_{-29}a, \ldots, X_{+30}t$) and features' indices (e.g. $0, 1, \ldots, 239$);
- injection layer: layer 0;
- knowledge: see slide 52;
- training: Adams as optimiser for 100 epochs (with early stop conditions);

Case study VI



Next in Line...

SKE & SKI


Sub-symbolic Data
Pre-Processing
Predictor Selection
SKI
Symbolic Knowledge
Training
Fixing
Predictions
Inference
Validation
SKE

Multi-Agent Systems

- agent to agent explanation [Omicini, 2020]
  $\rightarrow$ SKE + SKI + explanation;
- logic as lingua franca for communication between heterogeneous entities;
- knowledge sharing and knowledge exploitation among agents;
- symbolic techniques integrated with sub-symbolic ones
  $\rightarrow$ representing and manipulating cognitive processes and their results.

SKI: Symbolic Knowledge Injection

state of the art and research perspectives

Matteo Magnini
matteo.magnini@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum – Università di Bologna

07-06-2022

References I

[Bader et al., 2005] Bader, S., Garcez, A. S. d., and Hitzler, P. (2005).
Computing first-order logic programs by fibring artificial neural networks.
In Russell, I. and Markov, Z., editors, *Proceedings of the 18th International Florida Artificial Intelligence Research Society Conference (FlAIRS), Clearwater Beach, Florida, USA, May 15–17, 2005*, pages 314–319. AAAI Press
http://www.aaai.org/Library/FLAIRS/2005/flairs05-052.php.

[Baldi and Sadowski, 2016] Baldi, P. and Sadowski, P. J. (2016).
A theory of local learning, the learning channel, and the optimality of backpropagation.
Neural Networks, 83:51–74
DOI:10.1016/j.neunet.2016.07.006.

[Besold et al., 2017] Besold, T. R., d'Avila Garcez, A. S., Bader, S., Bowman, H., Domingos, P. M., Hitzler, P., Kühnberger, K., Lamb, L. C., Lowd, D., Lima, P. M. V., de Penning, L., Pinkas, G., Poon, H., and Zaverucha, G. (2017).
Neural-symbolic learning and reasoning: A survey and interpretation.
CoRR, abs/1711.03902
http://arxiv.org/abs/1711.03902.

References II

[Calegari et al., 2020] Calegari, R., Ciatto, G., and Omicini, A. (2020).
On the integration of symbolic and sub-symbolic techniques for XAI: A survey.
Intelligenza Artificiale, 14(1):7–32
DOI:10.3233/IA-190036.

[d'Avila Garcez and Gabbay, 2004] d'Avila Garcez, A. S. and Gabbay, D. M. (2004).
Fibring neural networks.
In McGuinness, D. L. and Ferguson, G., editors, *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, pages 342–347. AAAI Press / The MIT Press
http://www.aaai.org/Library/AAAI/2004/aaai04-055.php.

[Gunning, 2016] Gunning, D. (2016).
Explainable artificial intelligence (XAI).
Funding Program DARPA-BAA-16-53, Defense Advanced Research Projects Agency (DARPA)
http://www.darpa.mil/program/explainable-artificial-intelligence.

References III

[Omicini, 2020] Omicini, A. (2020).
Not just for humans: Explanation for agent-to-agent communication.
In Vizzari, G., Palmonari, M., and Orlandini, A., editors, *AIxIA 2020 DP — AIxIA 2020 Discussion Papers Workshop*, volume 2776 of *AI*IA Series*, pages 1–11, Aachen, Germany. Sun SITE Central Europe, RWTH Aachen University
http://ceur-ws.org/Vol-2776/paper-1.pdf.

[Towell and Shavlik, 1994] Towell, G. G. and Shavlik, J. W. (1994).
Knowledge-based artificial neural networks.
Artif. Intell., 70(1-2):119–165
DOI:10.1016/0004-3702(94)90105-8.

[van Gelder, 1990] van Gelder, T. (1990).
Why distributed representation is inherently non-symbolic.
In Dorffner, G., editor, *Konnektionismus in Artificial Intelligence und Kognitionsforschung. Proceedings 6. Österreichische Artificial Intelligence-Tagung (KONNAI), Salzburg, Österreich, 18. bis 21. September 1990*, volume 252 of *Informatik-Fachberichte*, pages 58–66. Springer
DOI:10.1007/978-3-642-76070-9_6.

References IV

[Wang et al., 2017] Wang, Q., Mao, Z., Wang, B., and Guo, L. (2017).
Knowledge graph embedding: A survey of approaches and applications.
IEEE Trans. Knowl. Data Eng., 29(12):2724–2743
DOI:10.1109/TKDE.2017.2754499.

[Xie et al., 2019] Xie, Y., Xu, Z., Meel, K. S., Kankanhalli, M. S., and Soh, H. (2019).
Embedding symbolic knowledge into deep networks.
In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4235–4245
https://proceedings.neurips.cc/paper/2019/hash/7b66b4fd401a271a1c7224027ce111bc-Abstract.html.