INTEGRATION OF LEARNING AND REASONING

Summary of some reading material

Temur Kutsia
RISC, Johannes Kepler University Linz

JOHANNES KEPLER
UNIVERSITY LINZ

Content

A brief summary of the chapter:

- Robin Manhaeve, Giuseppe Marra, Thomas Demeester, Sebastijan Dumancic, Angelika Kimmig, Luc De Raedt:

  Neuro-Symbolic AI = Neural + Logical + Probabilistic AI.

  In: Pascal Hitzler, Md. Kamruzzaman Sarker: Neuro-Symbolic Artificial Intelligence: The State of the Art. Frontiers in Artificial Intelligence and Applications 342, IOS Press 2021.

Plus some input from the recent survey:

- Giuseppe Marra, Sebastijan Dumancic, Robin Manhaeve, Luc De Raedt: From statistical relational to neurosymbolic artificial intelligence: A survey. Artif. Intell. 328: 104062 (2024)

Content

A brief summary of the chapter:

- Robin Manhaeve, Giuseppe Marra, Thomas Demeester, Sebastijan Dumancic, Angelika Kimmig, Luc De Raedt:

  Neuro-Symbolic AI = Neural + Logical + Probabilistic AI.

  In: Pascal Hitzler, Md. Kamruzzaman Sarker: Neuro-Symbolic Artificial Intelligence: The State of the Art. Frontiers in Artificial Intelligence and Applications 342, IOS Press 2021.

Plus some input from the recent survey:

- Giuseppe Marra, Sebastijan Dumancic, Robin Manhaeve, Luc De Raedt: From statistical relational to neurosymbolic artificial intelligence: A survey. Artif. Intell. 328: 104062 (2024)

Disclaimer: I do not claim that I understood everything.

Neuro-Symbolic AI

Integration of learning and reasoning: one of the key challenges in AI.

This chapter explores the integration of learning and reasoning in two different fields of artificial intelligence:

- neurosymbolic systems (NeSy, extending neural networks with logical reasoning)
- statistical relational artificial intelligence (StarAI, integrating logic with probabilistic graphical models)

and proposes putting the three base paradigms (logical, probabilistic, and neural methods) together for neuro-symbolic AI.

Neuro-Symbolic AI = Neural + Logical + Probabilistic AI

Outline

1. Base paradigms
   - ☐ logical methods
   - ☐ Probabilistic methods
   - ☐ Neural methods
2. Integration
   - ☐ StarAI
   - ☐ NeSy
3. Neural Probabilistic Logic Programming
4. Challenges

BASE PARADIGMS

Base paradigms: logical methods

Clausal fragments of

- propositional logic
- relational logic (first-order logic without nonconstant function symbols)
- first-order logic

Clausal form: $h_1 \vee \cdots \vee h_k \leftarrow b_1 \wedge \cdots \wedge b_n$, where the h's and b's are atomic formulas.

All variables are implicitly universally quantified.

Definite clause: $k = 1$.   Fact: $k = 1,\ n = 0$.

The chapter focuses on logic programs: sets of definite clauses.

Base paradigms: logical methods

Example (Alarm, first-order form)

```
burglary.
earthquake.
at_home(mary).
at_home(john).

alarm ← earthquake.
alarm ← burglary.
calls(X) ← alarm, at_home(X).
```

Base paradigms: logical methods

Definite programs have least Herbrand model.

For the alarm example in first-order form, it is the set of ground atoms:

```
{   burglary, earthquake, alarm,
    at_home(mary), at_home(john),
    calls(mary), calls(john)    }
```

Every element of the least Herbrand model can be proved from the program by SLD-resolution (backward chaining).

Programs containing general clauses are not guaranteed to have the least Herbrand model: there are several minimal ones.

Base paradigms: probabilistic methods

Probabilistic graphical models are graph structures, where the nodes represent random variables, existing edges represent dependences between them, and missing edges indicate independence.

Base paradigms: probabilistic methods

Probabilistic graphical models are graph structures, where the nodes represent random variables, existing edges represent dependences between them, and missing edges indicate independence.

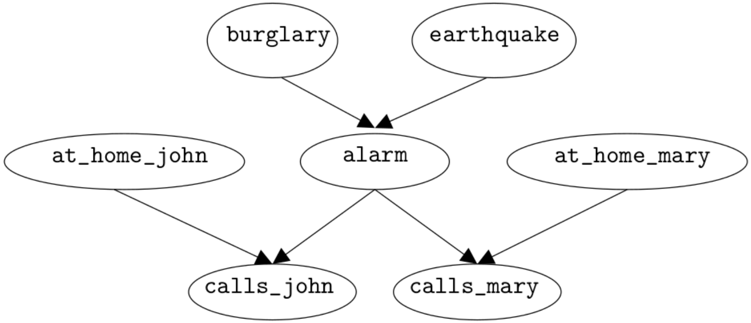
Two main classes of probabilistic graphical models:

- Bayesian networks, where the structure is a directed acyclic graph, and
- Markov random fields, where the graph is undirected.

Bayesian networks model asymmetric (causal) effects and dependencies.

The chapter focuses on Bayesian networks and their connections with definite logic programs.

Base paradigms: probabilistic methods



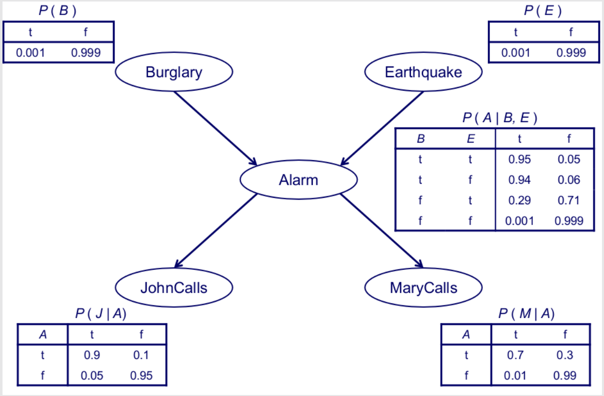
Base paradigms: probabilistic methods

Bayesian belief networks compactly represent (joint) probability distribution $P(X_1, \ldots, X_n)$ over n random variables $X_1, \ldots, X_n$ using the formula

$$P(X_1, \ldots, X_n) = \Pi_{i=1}^{n} P(X_i \mid parents(X_i)).$$

Base paradigms: probabilistic methods

Example (Simpler network, computing a joint entry)



$P(B)$

t	f
0.001	0.999

$P(E)$

t	f
0.001	0.999

$P(A \mid B, E)$

B	E	t	f
t	t	0.95	0.05
t	f	0.94	0.06
f	t	0.29	0.71
f	f	0.001	0.999

$P(J \mid A)$

A	t	f
t	0.9	0.1
f	0.05	0.95

$P(M \mid A)$

A	t	f
t	0.7	0.3
f	0.01	0.99

$$P(J, M, A, \neg B, \neg E) =$$
$$P(J|A)P(M|A)P(A|\neg B, \neg E)P(\neg B)P(\neg E) =$$
$$0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.999 = 0.000628741$$

Base paradigms: neural methods

Neural networks excel in machine learning tasks where the input data is high-dimensional and feature engineering is hard.

Examples: analyzing images, video or audio data, or natural language.

Combination of two principles:

- learning multiple layers of nonlinear functions that project the input data onto relatively low-dimensional embedding spaces,
- efficient training of these nonlinear models by using of advanced optimization techniques.

Base paradigms: neural methods

In classification tasks, neural networks typically learn a conditional probability distribution $P(Y|X)$, where Y is a random variable representing the class and X is the input pattern.

For example, a neural network can model the probability $P(Y = 2|X =$ $)$ that the MNIST image represents 2.

Neural networks can be seen as compact approximations of conditional probability distribution tables.

They in general perform badly on reasoning tasks.

Base paradigms: neural methods

Side remark: The MNIST database (Modified NIST database) is a large database of handwritten digits.

Commonly used for training various image processing systems.

INTEGRATION

Integration: StarAI

Statistical Relational AI.

Extends first-order logic (programming) with uncertainty.

Or, equivalently, extends probability theory with relations.

One key idea: to unify the notion of a logical atom and a random variable.

Integration: StarAI

StarAI models allow defining templated probabilistic models thanks to the use of first order clauses with variables.

The parameters of these models can be trained on a specific set of individuals (e.g. `mary` and `john`)) and then they can be used to make predictions on new individuals (e.g. `bob`).

ProbLog: a probabilistic logic programming language.

Integration: StarAI

ProbLog extends Prolog with probabilistic facts: $p :: f$.

The alarm example as a ProbLog program:

Example (Alarm, ProbLog)

```
0.1 :: burglary.
0.2 :: earthquake.
0.5 :: at_home(mary).
0.4 :: at_home(john).

alarm ← earthquake.
alarm ← burglary.

calls(X) ← alarm, at_home(X).
```

Integration: StarAI

A ProbLog program consists of a set of probabilistic facts and a set of definite clauses.

Each ground instance $f\vartheta$ of a probabilistic fact $p :: f$ corresponds to an independent Boolean random variable that is true with probability p and false with probability $1 - p$.

Integration: StarAI

A ProbLog program consists of a set of probabilistic facts and a set of definite clauses.

Each ground instance $f\vartheta$ of a probabilistic fact $p :: f$ corresponds to an independent Boolean random variable that is true with probability p and false with probability $1 - p$.

Given a ProbLog program $\mathcal{P}$, denote:

- $\mathcal{F}$: the set of probabilistic facts in $\mathcal{P}$,
- Ground$(\mathcal{F})$: the set of all ground instances of facts in $\mathcal{F}$,
- $\mathcal{R}$: the set of rules in $\mathcal{P}$.

Integration: StarAI

Every subset $F \subseteq \textsc{Ground}(\mathcal{F})$ defines a possible world

$$w_F = \{h \mid F \cup \mathcal{R} \models h \text{ and } h \text{ is ground}\}.$$

w_F is the least Herbrand model of the definite program $F \cup \mathcal{R}$.

Example

$F = \{$`burglary`, `at_home(mary)`$\}$.

```
0.1 :: burglary.
0.5 :: at_home(mary).
alarm ← earthquake.
alarm ← burglary.
calls(X) ← alarm, at_home(X).
```

$w_F = F \cup \{$`alarm`, `calls(mary)`$\}$.

Integration: StarAI

The probability $P(w_F)$ of such a possible world w_F:

$$P(w_F) = \prod_{f_i \in F} p_i \prod_{f_i \in \textsc{Ground}(\mathcal{F}) \setminus F} (1 - p_i)$$

Example

$F = \{$`burglary`, `at_home(mary)`$\}$.

```
0.1 :: burglary.              alarm ← earthquake.
0.2 :: earthquake.            alarm ← burglary.
0.5 :: at_home(mary).
0.4 :: at_home(john).         calls(X) ← alarm, at_home(X).
```

$w_F = \{$`burglary`, `at_home(mary)`, `alarm`, `calls(mary)`$\}$.

$$P(w_F) = 0.1 \times 0.5 \times (1 - 0.2) \times (1 - 0.4) = 0.024$$

Integration: StarAI

The probability of a ground atom q (success probability of q), is defined as the sum of the probabilities of all worlds containing q:

$$P(q) := \sum_{F \subseteq \textsc{Ground}(\mathcal{F}),\, q \in w_F} P(w_F).$$

To compute $P(\texttt{alarm})$, we should consider $P(w_F)$'s for 12 F's:

$$F_1 = \{\texttt{burglary}\}$$
$$F_2 = \{\texttt{burglary},\ \texttt{at\_home(mary)}\}$$
$$F_3 = \{\texttt{burglary},\ \texttt{at\_home(john)}\}$$
$$F_4 = \{\texttt{burglary},\ \texttt{at\_home(mary)},\ \texttt{at\_home(john)}\}$$
$$F_5 = \{\texttt{earthquake}\}$$
$$\ldots$$
$$F_{12} = \{\texttt{burglary},\ \texttt{earthquake},\ \texttt{at\_home(mary)},\ \texttt{at\_home(john)}\}.$$

Integration: StarAI

ProbLog allows many shortcuts for making programming easier (non-ground probabilistic facts, annotated disjunctions, probabilistic clauses, ...).

Bayesian networks can be encoded as ProbLog programs.

Integration: StarAI

ProbLog allows many shortcuts for making programming easier (non-ground probabilistic facts, annotated disjunctions, probabilistic clauses, …).

Bayesian networks can be encoded as ProbLog programs.

Annotated disjunction to model different severities of the earthquake:

```
0.4 :: no_earthquake ;
0.4 :: mild_earthquake ;
0.2 :: severe_earthquake.
```

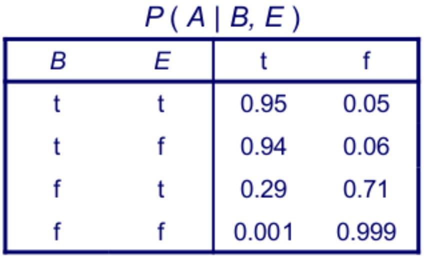
Or without explicitly representing the event of no earthquake:

```
0.4 :: mild_earthquake ; 0.2 :: severe_earthquake.
```

in which “neither `mild_earthquake` nor `severe_earthquake`” will be true with probability 0.4.

Integration: StarAI

Conditional probability distribution table as a ProbLog program:



$P(A \mid B, E)$

B	E	t	f
t	t	0.95	0.05
t	f	0.94	0.06
f	t	0.29	0.71
f	f	0.001	0.999

```
0.95  :: alarm :- burglary, earthquake.
0.94  :: alarm :- burglary, \+ earthquake.
0.29  :: alarm :- \+ burglary, earthquake.
0.001 :: alarm :- \+ burglary, \+ earthquake.
```

Integration: NeSy

Neuro-symbolic AI integrates neural networks with symbolic representations.

Two types of neuro-symbolic AI:

- neural networks extended with logical aspects,
- logical approaches extended with neural constructs.

Integration: NeSy

Neural networks extended with logical aspects. One way:

- logical constraints are imposed on the output,
- not hard constraints (not guaranteed to hold),
- only used during the training, where the degree in which they are not satisfied serves as an additional loss term,
- enforces the neural methods to make predictions that adhere better to the logic,
- encodes the logic into the parameters of the network, so that even when the logic is not explicitly present, the model should still satisfy the logical constraints more than models that were trained without these constraints.

Examples: semantic based regularization, the semantic loss function.

Integration: NeSy

Neural networks extended with logical aspects. Another way:

- using a carefully designed architecture (e.g., as in neuro-symbolic deductive reasoning or deep deductive reasoning), or
- using the logic to define the neural network structure in a templating approach (e.g., relational neural networks, neural theorem provers, etc.).

Integration: NeSy

Logical methods with neural constructs.

The neural constructs create an interface between the logic-based framework and the neural network.

These constructs allow the logic to evaluate neural networks such that their parameters can be optimized together with any possible parameters in the logic.

This type of extension is similar to how Prolog was extended into ProbLog through the introduction of probabilistic facts.

Integration: NeSy

Numbers vs Booleans.

Neuro-symbolic AI systems need to connect the output of the neural networks to the Boolean values in logic.

Integration: NeSy

Numbers vs Booleans.

Neuro-symbolic AI systems need to connect the output of the neural networks to the Boolean values in logic.

Seeing neural networks as classifiers, they output a confidence score for each class.

Integration: NeSy

Numbers vs Booleans.

Neuro-symbolic AI systems need to connect the output of the neural networks to the Boolean values in logic.

Seeing neural networks as classifiers, they output a confidence score for each class.

If the neural network is sufficiently confident, it might suffice to only select the top prediction.

However, in general, this is not feasible.

There are several strategies for dealing with this.

Integration: NeSy

Numbers vs Booleans, one strategy.

Choose one of the outputs of the neural network, based on the confidence score.

When the choices turn out to be logically inconsistent, different outputs are chosen until they are consistent.

These choices can be used train the neural network to make this output more likely.

Used in systems ABL, NGS, . . .

Integration: NeSy

Numbers vs Booleans, another strategy.

Use the confidence scores directly in the logic, turning logical operators into real-valued functions

This relaxes Boolean truth values to the continuous $[0, 1]$ interval.

This introduces T-norm-based fuzzy logics.

Used in systems LRNN, DiffLog, …

Integration: NeSy

Numbers vs Booleans, yet another strategy.

Interpret the confidence scores as a probability distribution.

Use a probabilistic logic to deal with the uncertainty.

Used in systems DeepProbLog, NeurASP, …

NEURAL PROBABILISTIC LOGIC PROGRAMMING

Neural Probabilistic Logic Programming

The authors approach: two desirable properties of frameworks that integrate two other frameworks A and B:

1. The original frameworks A and B should be a special case of the integrated one.
2. Models that learn from observed samples should be able to deal with uncertainty.

Neural Probabilistic Logic Programming

The authors approach: two desirable properties of frameworks that integrate two other frameworks A and B:

1. The original frameworks A and B should be a special case of the integrated one.
2. Models that learn from observed samples should be able to deal with uncertainty.

The authors claim that the first property is not satisfied by the vast majority of neuro-symbolic approaches.

The second property implies that one should not only integrate logic with neural networks in NeSy, but also probability.

Neuro-Symbolic AI = Neural + Logical + Probabilistic AI.

Neural Probabilistic Logic Programming

DeepProbLog: neuro-symbolic extension of ProbLog.

In ProbLog, the probabilities of all random choices are explicitly specified as part of probabilistic facts or annotated disjunctions.

Neural Probabilistic Logic Programming

DeepProbLog: neuro-symbolic extension of ProbLog.

In ProbLog, the probabilities of all random choices are explicitly specified as part of probabilistic facts or annotated disjunctions.

DeepProbLog extends ProbLog to basic random choices whose probabilities are parameterized by neural networks.

This is realized through neural predicates: special probabilistic facts that are annotated by neural networks instead of by scalar probabilities.

Neural Probabilistic Logic Programming

Neural predicate is used to define a neural annotated disjunction (nAD).

It is assumed that a neural network model m is given.

The network model m defines a probability distribution

$$p_m(Y = t \mid X_1 = s_1, \ldots, X_k = s_k)$$

where

- $t \in \{t_1, \ldots, t_n\}$ (t_i's are ground terms, $\{t_1, \ldots, t_n\}$ is the output domain),
- $s_1, \ldots, s_k$ represent the input to the neural network.

Neural Probabilistic Logic Programming

Example

This nAD specifies probability distribution for MNIST digits

```
nn(m_digit, [X], Y, [0,...,9]) :: digit(X, Y).
```

where `m_digit` is a network that classifies MNIST digits.

For input image , this generic nAD represents a ground nAD containing 10 disjuncts:

p_{m_digit}(Y = 0 | X =) :: digit(, 0) ;

...

p_{m_digit}(Y = 9 | X =) :: digit(, 9)

Note that DeepProbLog allows images or other subsymbolic representations as terms of the program.

Neural Probabilistic Logic Programming

DeepProbLog: how does it work.

Program:

```
nn(m_digit, [X], Y, [0,...,9]) :: digit(X, Y).
addition(X,Y,Z) :-
    digit(X,N1), digit(Y,N2), Z is N1+N2.
```

Query: `addition(0, 1, 1)`

Step 1: grounding the program (relevant part)

```
nn(m_digit, [0], 0) :: digit(0, 0) ;
nn(m_digit, [0], 9) :: digit(0, 9).
nn(m_digit, [1], 0) :: digit(1, 0) ;
nn(m_digit, [1], 9) :: digit(1, 9).
addition(0,1,1) :- digit(0,0), digit(1,1).
addition(0,1,1) :- digit(0,1), digit(1,0).
```

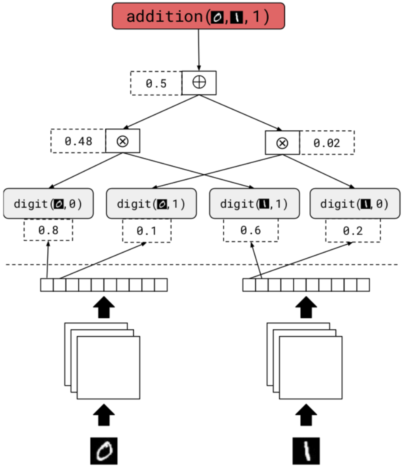
Neural Probabilistic Logic Programming

Step 2: obtain form the ground program a propositional formula that defines the truth value of the query in terms of the truth values of probabilistic facts:

(digit(,0) ∧ digit(,1)) ∨
(digit(,1) ∧ digit(,0)).

Neural Probabilistic Logic Programming

Step 3: transform the obtained formula into an arithmetic circuit and use it to compute the success probability of the query.



CHALLENGES

Challenges for neurosymbolic AI

- Scalability
  - Can neural networks help? Or approximate inference?
- Structure learning
  - Currently, providing clause templates or a model sketch is required
- Semantics
  - Many competing formalisms.
- Data efficiency
  - StarAI is can learn from little data set, but does not scale. Neural networks require large data sets.
- Symbolic representation learning
  - Is it possible to change the representation at the symbolic level like deep learning does to the data to simplify solving the target task?

Frameworks	Inference	Syntax	Semantics	Learning	Representations	Paradigms	Tasks
	(P)roof (M)odel	(P)ropositional (R)elational (FOL)	(M)inimal (S)table (C)lassical (F)uzzy (P)robability	(P)arameters (S)tructure	(S)ymbolic (Sub)symbolic	Logic (L/l) Probability (P/p) Neural(N/n)	(D)istant (S)upervision (S)emi (S)upervised (KGC)ompletion (G)enerative (K)nowledge (I)nduction
αILP [111]	P+M	FOL	S + P	P + S	S	Ln	KI
∂ILP [39]	P	R	M + F	P + S	S	Ln	DS + KI
DeepProbLog [72]	P+M	FOL	M + P	P+S	S+Sub	LpN	DS + KI
DeepStochLog [132]	P	FOL	M + P	P	S	LpN	DS + SS
DiffLog [112]	P	R	M + F	P+S	S	Ln	KI
DL2 [40]	M	P	C + F	P	S+Sub	lN	DS + SS
DLM [77]	M	FOL	C + F + P	P	S	lPN	SS + KGC
LRNN [116]	P	R	M + F	P + S	S + Sub	LN	KGC + KI
LTN [5]	M	FOL	C + F	P	S + Sub	lN	DS + SS
NeuralLP [137]	P	R	M + F	P	S	Ln	KGC + KI
NeurASP [138]	P+M	FOL	S + P	P	S	LpN	DS
NLM [35]	P	R	M + F	P + S	S	Ln	KGC + KI
NLog [121]	P	R	M + P	P	S	LpN	DS
NLProlog [131]	P	R	M + P	P + S	S + Sub	LpN	KGC + KI
NMLN [78]	M	FOL	C + P	P + S	S + Sub	lPN	KGC + G
NTP [102]	P	R	M + F	P + S	S + Sub	Ln	KGC + KI
RNM [76]	M	FOL	C + P	P	S	lPN	SS
SBR [33]	M	FOL	C + F	P	S+Sub	lN	DS + SS
Scallop [58]	P	FOL	M + P	P	S	LpN	DS
SL [133]	M	P	C + P	P	S	LpN	SS
Slash [113]	P+M	FOL	S + P	P	S	LpN	DS +SS
TensorLog [18]	P	R	M + P	P	S	LpN	DS + KGC