Explainable Artificial Intelligence

03 – Neuro-symbolic approaches

MSc in Artificial Intelligence

MSc in Computer Engineering

Marco Lippi

marco.lippi@unifi.it

UNIVERSITÀ DEGLI STUDI FIRENZE

UNIVERSITÀ DEGLI STUDI FIRENZE

Table of Contents

1 General concepts

The connectionist vs. symbolic dilemma

A central question in AI: how is knowledge represented in our mind?

Symbolic approaches

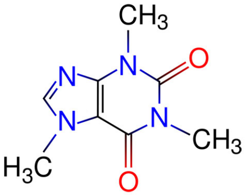
- Reasoning as the result of formal manipulation of symbols

Connectionist (sub-symbolic) approaches

- Reasoning as the result of processing of interconnected (networks of) simple units

Symbolic approaches

- Founded on the **principles of logic**
- Exploiting **background knowledge**
- Highly **interpretable**
- Need **discretization**



toxic(M) :- doublebond(M,C1,C2), hydroxyl(C2), methyl(M)

Connectionist vs. symbolic AI

Connectionist approaches

- Can more easily deal with **uncertain knowledge**
- No need to **encode** knowledge at all!
- Need to observe (large) **data** collections!
- Algorithms typically can be easily **distributed**
- Often seen as “black box” $\rightarrow$ dark magic
- **Graceful degradation** of performance with noisy input

Connectionist vs. symbolic AI

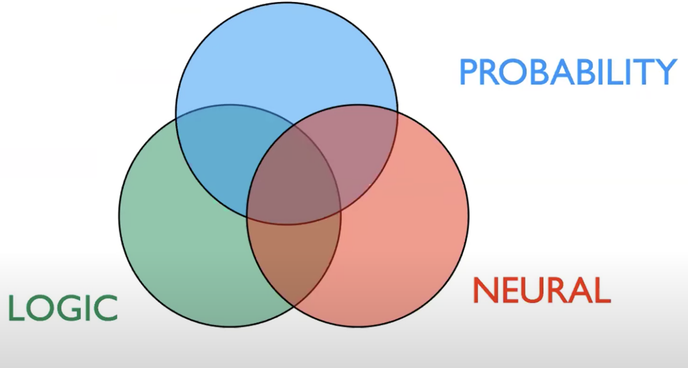
Both a technological and a philosophical difference...

- Do we (as humans) really need huge data collections to learn?
- How do we acquire **generalization** skills?
- Can we easily define **rules** that drive our decisions?
- Can we easily describe data through a symbolic formalism?

Table of Contents

2 Combining learning and reasoning

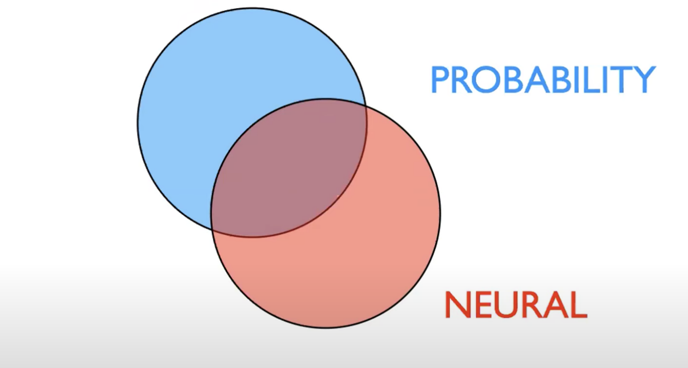
Combining learning and reasoning



[Image by Luc De Raedt]

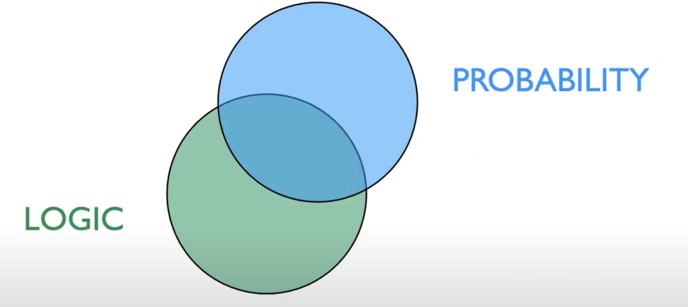
Combining learning and reasoning

Machine (and deep) learning



[Image by Luc De Raedt]

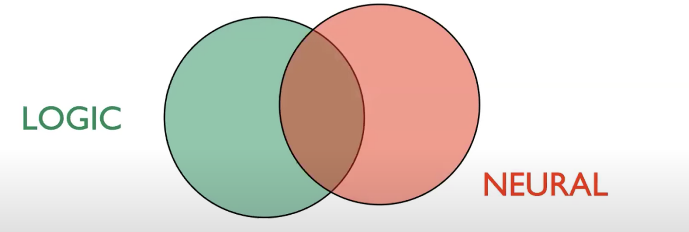
Statistical Relational Learning (SRL/StarAI)



[Image by Luc De Raedt]

Combining learning and reasoning

Neural Symbolic Computation (NeSy)

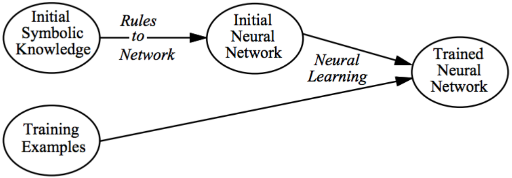


[Image by Luc De Raedt]

Pioneering approaches: KBANNs

Knowledge-based artificial neural networks [Towell & Shavlik, 1994]

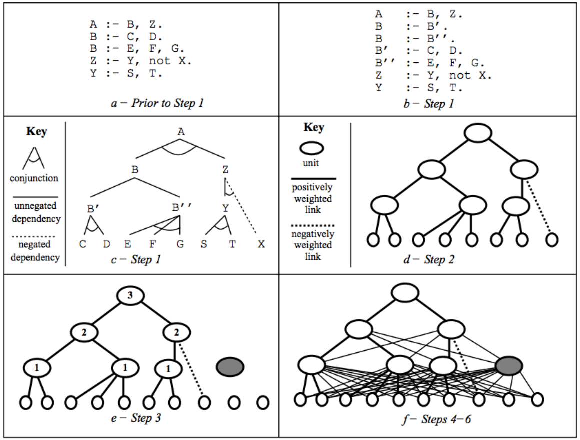
- One of the first attempts to inject knowledge into ANNs
- Trying to interpret an ANN model as logic rules



Knowledge Base		*Neural Network*
Final Conclusions	$\Longleftrightarrow$	Output Units
Supporting Facts	$\Longleftrightarrow$	Input Units
Intermediate Conclusions	$\Longleftrightarrow$	Hidden Units
Dependencies	$\Longleftrightarrow$	Weighted Connections

Pioneering approaches: KBANNs

2 Combining learning and reasoning



NeSy and SRL/StarAI

More recent research directions:

- **Statistical Relational Learning** (SRL) or **Statistical Relational AI** (StarAI)
- **Neural-Symbolic Learning and Reasoning** (NeSy)

Both developed during the 90s-00s
The former combining logic with probabilistic/statistical learning (SRL/StarAI)
The latter combining logic with cognitive neuroscience (NeSy)

Which logic?
Which probabilistic models?
Which neural models?

SRL/StarAI

Aiming to combine **first-order logic** and **graphical models** for learning and reasoning

- Exploit the **expressive power** of first-order logic
- Handle uncertainty with **graphical models**
- Combine **logic and probabilistic** inference

Aiming to combine **neural models** and **symbolic approaches** for learning and reasoning

- Encode knowledge in the **architecture** of the network
- Use a **regularization** term to encode rules
- **Constrain** neural computations with rules

Caveat: injecting knowledge into the neural network, then let the network do the rest might not be sufficient (partly lost the power of reasoning and explanation)

Logic can be used in several ways [De Raedt et al., 2020]:

- Logic as a kind of neural program
- Logic as a regularizer
- ...

Logic as **a kind of neural program**

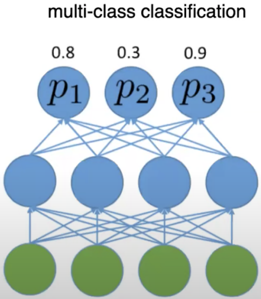
Logic cabled within the architecture of the network
This comes from the idea of KBANNs by Towell and Shavlik...

Logic as **regularizer**

Combine standard classification loss with a loss that takes into account constraints, by penalizing solutions that break them...

$$\text{Loss} = \text{ClassificationLoss} + \lambda \cdot \text{SemanticLoss}$$

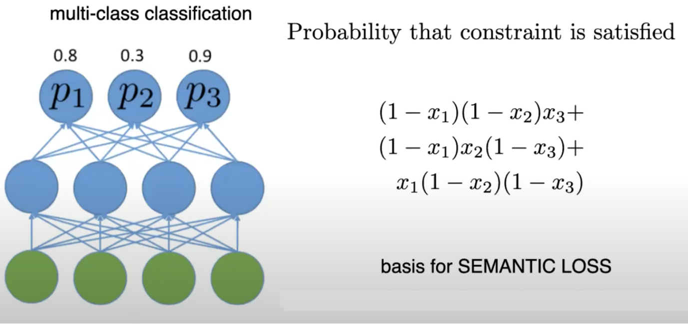
But... If logic is encoded in the network, how to **reason logically**?



This constraint should be satisfied

$$
\begin{aligned}
&(\neg x_1 \wedge \neg x_2 \wedge x_3) \vee \\
&(\neg x_1 \wedge x_2 \wedge \neg x_3) \vee \\
&(x_1 \wedge \neg x_2 \wedge \neg x_3)
\end{aligned}
$$

from Xu et al., ICML 2018



multi-class classification
0.8
0.3
0.9
p_1
p_2
p_3
Probability that constraint is satisfied
$(1-x_1)(1-x_2)x_3+$
$(1-x_1)x_2(1-x_3)+$
$x_1(1-x_2)(1-x_3)$
basis for SEMANTIC LOSS

Table of Contents

3 Tasks

Classic SRL/StarAI tasks

StarAI applications typically deal with three distinct, but strongly inter-related problems…

- Inference
- Parameter Learning
- Structure Learning

Inference

Inference in StarAI lies at the intersection between logical and probabilistic inference

Logical Inference
Inferring the truth value of some logic facts, given a collection of other facts and rules

Probabilistic inference
Inferring the posterior distribution of unobserved random variables, given observed ones

Logical inference

Logical theory

```
likesmovie(X,M) :- moviegenre(M,G), likesgenre(X,G).
likesmovie(X,M) :- friends(X,Y), likesmovie(Y,M).
```

Caveat: need of grounding out...

```
likesmovie(alice,bladerunner).
friends(alice,bob).
likesgenre(alice,sciencefiction).
...
```

Logical inference

Main techniques

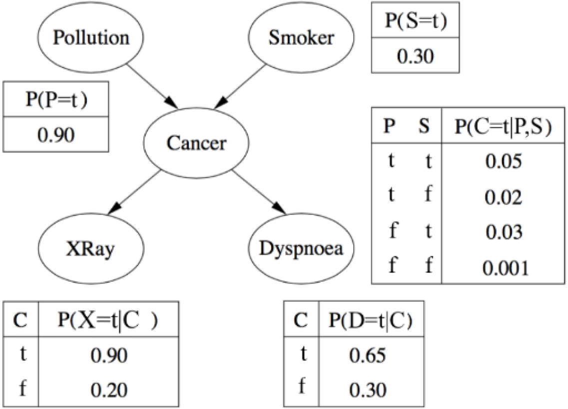
- Model-based, SAT solvers
- Proof-based, finding derivations

Grounding is in any case the **bottleneck**...

- Smart techniques to avoid generating **all** the predicates
- Exploit simmetries and patterns/templates

Probabilistic inference

An example in Bayesian Networks



P(S=t)
0.30

P(P=t)
0.90

P	S	P(C=t|P,S)
t	t	0.05
t	f	0.02
f	t	0.03
f	f	0.001

C	P(X=t|C)
t	0.90
f	0.20

C	P(D=t|C)
t	0.65
f	0.30

Parameter learning

Typically, StarAI models specify a set of **parameters** (probabilities or real values) attached to rules/clauses to model uncertainty in the available knowledge

These parameters can be **learned from data**:

- probability tables in Bayesian networks
- weights or probabilities attached to soft rules

Structure learning

Highly challenging problem: directly learning the rules (the **structure**) of the model

Different approaches...

- Jointly learn parameters and rules
- First learn rules (i.e., with ILP), then their weights

Collective classification

This framework could be easily exploited to perform collective classification on a set of **non-independent examples**, like nodes in a graph or sentences in a document, . . .

Given a set of (possibly neural) rules, and a collection of constants/features representing the document, the inference algorithm computes the **most likely world**, or interpretation, thus assigning a truth value to each predicate in the document.

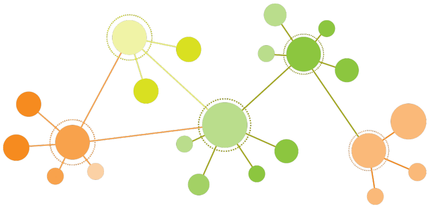


Table of Contents

4 Models and Frameworks

Markov Logic Networks [Richardson & Domingos, 2006]

Logic imposes **hard constraints** on the set of possible worlds
Markov logic exploits **soft constraints**

A Markov Logic Network is defined by:

- a set of first-order formulae
- a set of weights, one attached to each formula

A world violating a formula is less probable, **not** impossible!

Markov Logic Networks [Richardson & Domingos, 2006]

A probabilistic-logic framework to model knowledge

Syntax **opposite to Prolog**: constants uppercase, variables lowercase

An example

```
Movie = {BladeRunner, TheMatrix}
Person = {Alice, Bob, Carl, David}
2.3  LikesMovie(x,m) ∧ Friends(x,y) => LikesMovie(y,m)
1.6  Friends(x,y) ∧ Friends(y,z) => Friends(x,z)
```

The **higher** the weight, the **more likely** a world where rule is true, other things being equal

Markov Logic Networks [Richardson & Domingos, 2006]

Beware of the differences in the syntax:

- MLN: uppercase constants (e.g., Alice) and lowercase variables (e.g., person)
- ProbLog: lowercase constants (e.g., alice) and uppercase variables (e.g., Person)

The semantics of MLNs induces a **probability distribution** over all possible worlds. We indicate with X a set of random variables represented in the model, then we have:

$$P(X = x) = \frac{\exp(\sum_{F_i \in \mathcal{F}} w_i n_i(x))}{Z}$$

being $n_i(x)$ the number of true groundings of F_i in world x and Z is the partition function:

$$Z = \sum_{x \in \mathcal{X}} \exp\left(\sum_{F_i \in \mathcal{F}} w_i n_i(x)\right)$$

Markov Logic Networks [Richardson & Domingos, 2006]

Discriminative setting

Typically, some atoms are always **observed** (evidence X), while others are **unknown** at prediction time (query Y)

EVIDENCE	QUERY
Friends(Alice,Bob)	LikedMovie(Alice,TheMatrix)???
Friends(Alice,Carl)	LikedMovie(Alice,PulpFiction)???
WatchedMovie(Alice,PulpFiction)	LikedMovie(Bob,TheMatrix)???
WatchedMovie(Bob,PulpFiction)	LikedMovie(Bob,PulpFiction)???

Markov Logic Networks [Richardson & Domingos, 2006]

The **probability** of a world/configuration depends on the **weights** (w_i) and the number of **groundings** (n_i) of each formula (F_i):

$$P(Y = y|X = x) = \frac{\exp(\sum_{F_i \in \mathcal{F}} w_i n_i(x, y))}{Z_x}$$

Inference aims to find the **most probable** y given x:

$$y^* = \text{argmax}_y P(Y = y|X = x)$$

Markov Logic Networks [Richardson & Domingos, 2006]

Inference

Given set of known facts, weighted rules used to infer truth value of other (query) facts

```
LikesMovie(Alice,BladeRunner)
Friends(Alice,Bob)
Friends(Alice,Carl)

LikesMovie(Carl,BladeRunner)???
```

#P-complete problem $\Rightarrow$ approximate algorithms

MaxWalkSAT [1996], **stochastic local search** $\Rightarrow$ minimize the sum of unsatisfied clauses

Markov Logic Networks [Richardson & Domingos, 2006]

A common problem in (probabilistic) logical inference is that you typically need to **ground** your knowledge base into your predicates.

This can quickly become **unaffordable** due to both time and memory requirements...

...Now moving towards **lifted inference**!

Markov Logic Networks [Richardson & Domingos, 2006]

Learning

Both **weights** and **rules** themselves can be **learned** from a collection of predicate observations.

Maximize conditional log likelihood (CLL) of query predicates given evidence: **inference as subroutine**!

$$\frac{\partial}{\partial w_i} \log P(Y = y | X = x) = n_i - E_w[n_i]$$

Ground-Specific MLNs [Lippi & Frasconi, 2009]

An extension of MLNs that allows to embed neural networks to compute weights

A simple classification example

```
w(x)    HasFeatures(x,$f) => PositiveClass(x)
```

The weight `w(x)` is computed by a neural network using (any) set of features `$f` describing example `x`.

These are named **Ground-Specific MLNs**.

Ground-Specific MLNs [Lippi & Frasconi, 2009]

An example in structured text classification

```
 2.3   Features(X,$F1) => CategoryA(X)
-1.8   Features(X,$F1) => CategoryB(X)
 0.9   Features(Y,$F2) => CategoryA(Y)
-0.7   Features(Y,$F2) => CategoryB(Y)
 1.1   Features(X,$F1) ∧ Features(Y,$F2) => Link(X,Y)
+Inf   Link(X,Y) => CategoryA(X) ∧ CategoryB(Y)
```

Ground-specific weights are computed by neural networks.
Infinite weights correspond to **hard** constraints.

Ground-Specific MLNs [Lippi & Frasconi, 2009]

- Inference algorithms **do not change**
- Learning algorithms implement **gradient descent**

$$\frac{\partial P(y|x)}{\partial \theta_k} = \frac{\partial P(y|x)}{\partial w_i} \frac{\partial w_i}{\partial \theta_k}$$

where the **first** term is computed by MLN inference and the **second** term is computed by backpropagation

ProbLog [De Raedt et al., 2007]

ProbLog is a **probabilistic extension of Prolog** where probabilities can be attached to ground facts or rules.

DeepProbLog extends ProbLog by computing such probabilities with neural networks in a framework for probabilistic reasoning

- Necessary to know Pro(b)Log
- Cannot (yet) perform collective classification

ProbLog [De Raedt et al., 2007]

A **ProbLog** example (probabilistic logic program)

```
person(alice).
person(bob).
person(carl).
movie(bladerunner).
movie(titanic).
friends(alice,bob).
friends(bob,alice).
likes(bob,bladerunner).

0.2::friends(X,Y) :- person(X), person(Y).
0.5::likes(X,M) :- person(X), movie(M).
0.8::likes(Y,M) :- person(X), movie(M), likes(X,M), friends(X,Y).

query(likes(_,_)).
```

Demo: `https://dtai.cs.kuleuven.be/problog/editor.html`

MLN vs. ProbLog

Weights vs. probabilities

- In an MLN, the weight of formula F represents the log-odds between a world where F is true and a world where F is false, other things being equal
- In ProbLog we model directly the probability that a rule is true

DeepProbLog [Manhaeve et al., 2018]

4 Models and Frameworks