logically-consistent deep learning via probabilistic circuits

antonio vergari (he/him)

@tetraduzione

2nd Apr 2025 - **Neuro-explicit retreat** *Saarbruecken*

probabilistic circuits (PCs)

A grammar for tractable computational graphs

I. *A simple tractable function is a circuit*

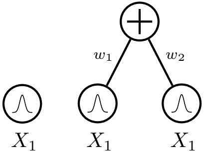
$\Longrightarrow$ *e.g., a multivariate Gaussian, or a logical literal*

probabilistic circuits (PCs)

A grammar for tractable computational graphs

I. *A simple tractable function is a circuit*

II. *A weighted combination of circuits is a circuit*



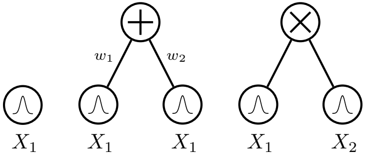
probabilistic circuits (PCs)

A grammar for tractable computational graphs

I. *A simple tractable function is a circuit*

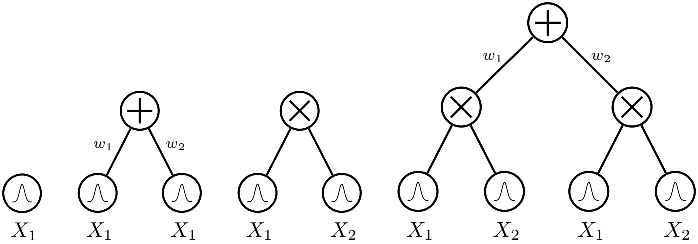
II. *A weighted combination of circuits is a circuit*

III. *A product of circuits is a circuit*



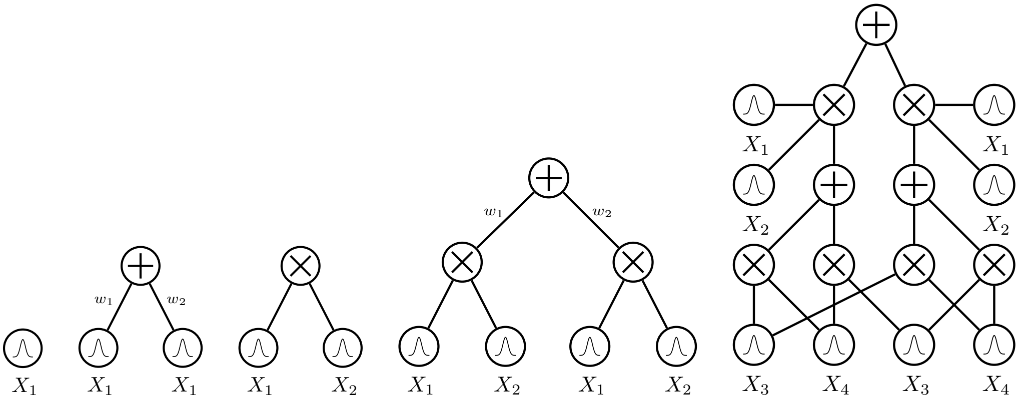
probabilistic circuits (PCs)

A grammar for tractable computational graphs



probabilistic circuits (PCs)

A grammar for tractable computational graphs



structural properties

smoothness

decomposability

compatibility

determinism

Vergari *et al., "A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference", NeurIPS, 2021*

structural properties

smoothness

decomposability

compatibility

determinism

Vergari *et al., "A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference", NeurIPS, 2021*

structural properties

smoothness

decomposability

compatibility

determinism

Vergari *et al., "A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference", NeurIPS, 2021*

structural properties

smoothness

decomposability

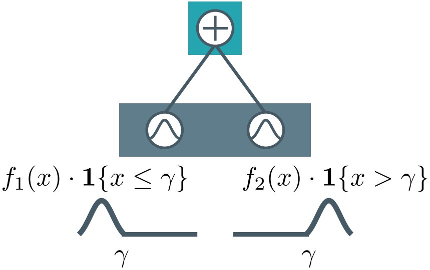
compatibility

determinism

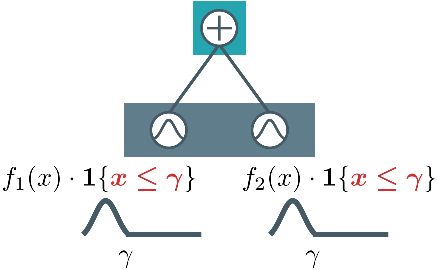
Vergari *et al., "A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference", NeurIPS, 2021*

determinism

the inputs of sum units are defined over disjoint supports

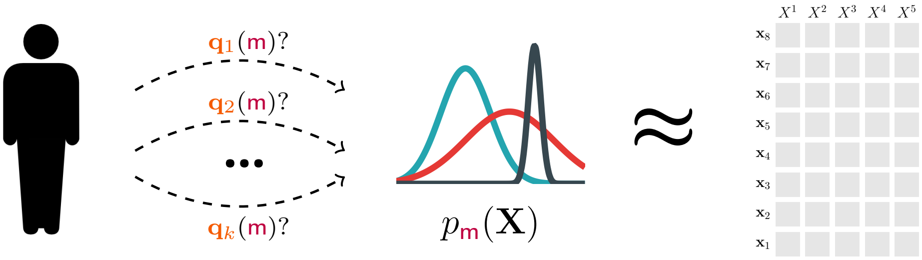


deterministic circuit

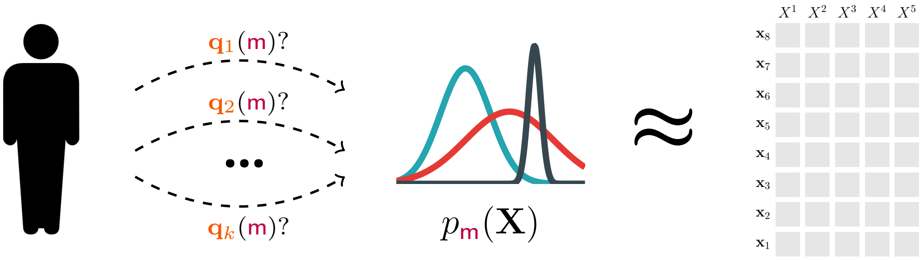


non-deterministic circuit

Darwiche and Marquis, "A knowledge compilation map", , 2002



generative models that can reason probabilistically



but some events should have zero probabilities!

Goal

"How can neural nets reason and learn with symbolic constraints reliably and efficiently?"

the issues!

I) Logical constraints can be hard to represent in a unified way

$\Longrightarrow$ ***a single framework*** *for implications, negation, paths, hierarchies, ...*

II) How to integrate logic and probabilities in a single architecture

$\Longrightarrow$ ***combining soft and hard constraints***

III) Logical constraints are piecewise constant functions!

$\Longrightarrow$ *differentiable almost everywhere but* ***gradient is zero****!*

the issues!

I) Logical constraints can be hard to represent in a unified way

$\Longrightarrow$ ***a single framework** for implications, negation, paths, hierarchies, ...*

II) How to integrate logic and probabilities in a single architecture

$\Longrightarrow$ ***combining soft and hard constraints***

III) Logical constraints are piecewise constant functions!

$\Longrightarrow$ *differentiable almost everywhere but **gradient is zero**!*

the issues!

I) Logical constraints can be hard to represent in a unified way

$\Longrightarrow$ ***a single framework*** *for implications, negation, paths, hierarchies, ...*

II) How to integrate logic and probabilities in a single architecture

$\Longrightarrow$ ***combining soft and hard constraints***

III) Logical constraints are piecewise constant functions!

$\Longrightarrow$ *differentiable almost everywhere but* ***gradient is zero****!*

hard vs soft constraints

logic vs probabilities

logic

"If X is a bird, X flies"

$$A(X) \implies B(X)$$

prob logic

"How likely is that if X is a bird, X flies?"

$$\mathsf{p}(A(X) \implies B(X))$$

which logic?

or which kind of constraints to represent?

propositional logic (zeroth-order)

$$(a \land b) \lor d \implies c$$

first-order logic (FOL)

$$\forall a \exists b : R(a, b) \lor Q(d) \implies C(x)$$

satisfiability modulo theory (SMT)

$$(\alpha X_i - \beta X_j \leq 100) \lor (X_j + X_k \geq 0) \implies (X_j X_k \leq X_i)$$

which logic?

or which kind of constraints to represent?

propositional logic (zeroth-order)

$$(a \wedge b) \vee d \implies c$$

first-order logic (FOL)

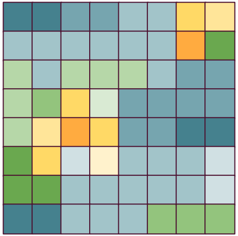
$$\forall a \exists b : R(a, b) \vee Q(d) \implies C(x)$$

satisfiability modulo theory (SMT)

$$(\alpha X_i - \beta X_j \leq 100) \vee (X_j + X_k \geq 0) \implies (X_j X_k \leq X_i)$$

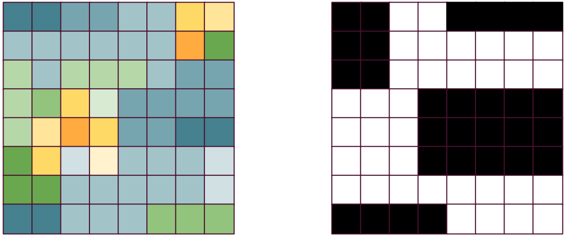
Goal

"How can we enforce symbolic constraints in a probability distribution?"



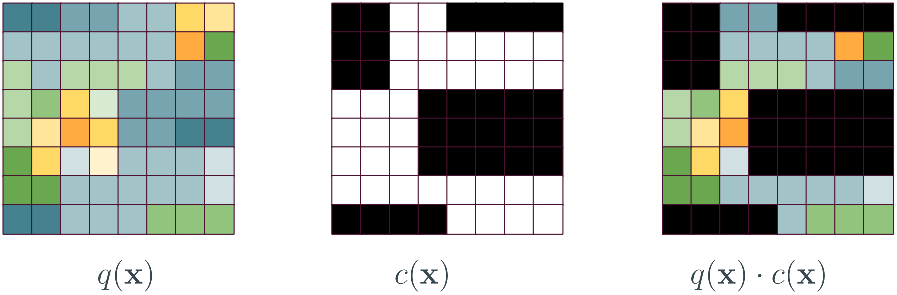
$q(\mathbf{x})$

start from a distribution $q(\mathbf{x})$**...**

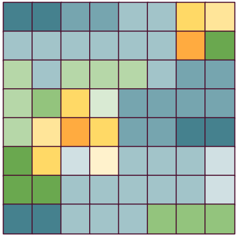


$q(\mathbf{x})$ $c(\mathbf{x})$

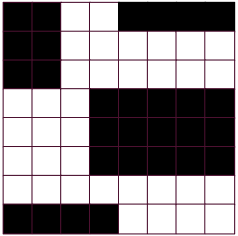
...and cut its support by a constraint $c(\mathbf{x})$



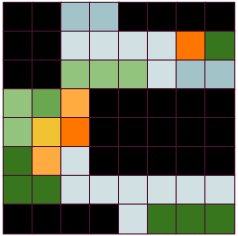
by multiplying them $q(\mathbf{x})c(\mathbf{x})$...



$q(\mathbf{x})$

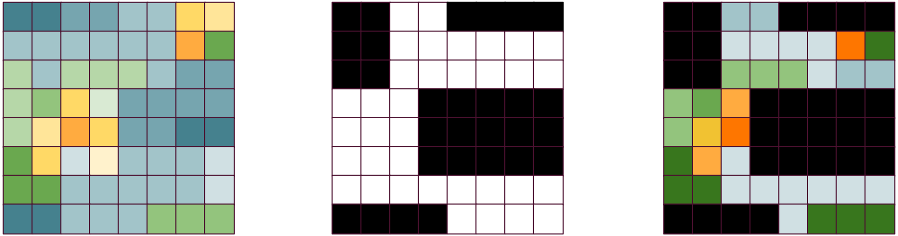


$c(\mathbf{x})$



$\frac{q(\mathbf{x})\cdot c(\mathbf{x})}{\sum_{\mathbf{x}} q(\mathbf{x})\cdot c(\mathbf{x})}$

and then renormalizing them!



$q(\mathbf{x})$ $c(\mathbf{x})$ $\frac{q(\mathbf{x})\cdot c(\mathbf{x})}{\sum_{\mathbf{x}} q(\mathbf{x})\cdot c(\mathbf{x})}$

states with zero probability will never be predicted (nor sampled)

Goal

Can we design q and c
to be expressive models
yet yielding a tractable product?

Goal

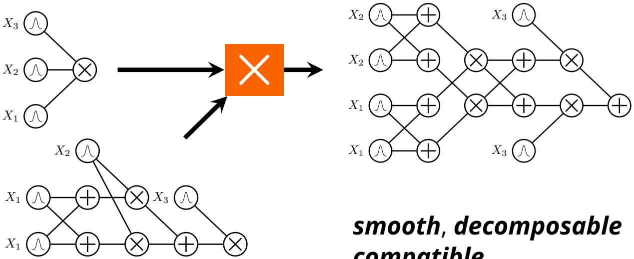
Can we design q and c to be deep computational graphs yet yielding a tractable product?

Goal

Can we design q and c
to be deep computational graphs
yet yielding a tractable product?

yes! as *circuits*!

Tractable products

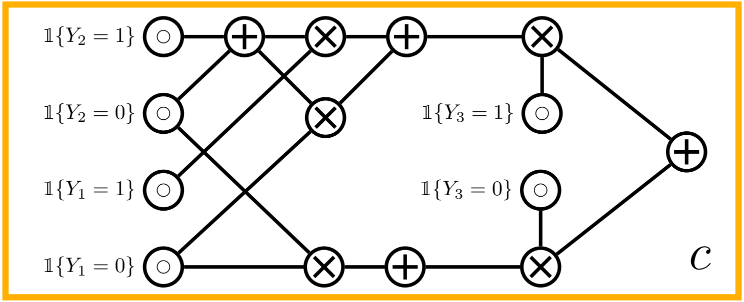


smooth, ***decomposable***
compatible

exactly compute $\mathcal{Z}$ in time $O(|\mathbf{q}||\mathbf{c}|)$

__Vergari__ et al., "A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference", NeurIPS, 2021

knowledge compilation



compiling logical formulas into circuits

(smooth, structured-decomposable, deterministic)

knowledge compilation

$$\mathsf{K}: (Y_1 = 1 \implies Y_3 = 1)$$

$$\land \quad (Y_2 = 1 \implies Y_3 = 1)$$

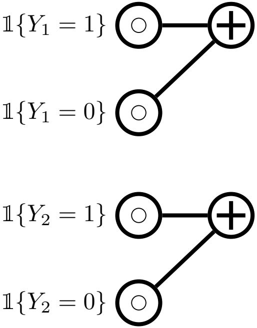
$\mathbb{1}\{Y_1 = 0\} \odot$ $\mathbb{1}\{Y_1 = 1\} \odot$ $\mathbb{1}\{Y_2 = 0\} \odot$

$\mathbb{1}\{Y_2 = 1\} \odot$ $\mathbb{1}\{Y_3 = 0\} \odot$ $\mathbb{1}\{Y_3 = 1\} \odot$

Pipatsrisawat and Darwiche, "New Compilation Languages Based on Structured Decomposability.", AAAI, 2008

knowledge compilation

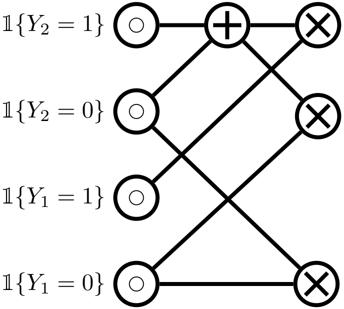
$$\mathsf{K} : (Y_1 = 1 \implies Y_3 = 1)$$
$$\wedge \quad (Y_2 = 1 \implies Y_3 = 1)$$



Pipatsrisawat and Darwiche, "New Compilation Languages Based on Structured Decomposability.", AAAI, 2008

knowledge compilation

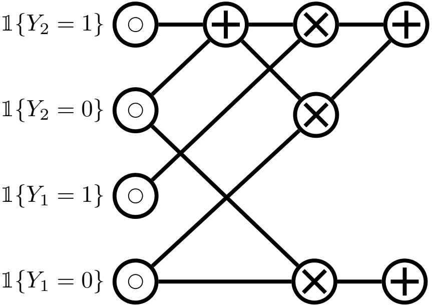
$$K : (Y_1 = 1 \implies Y_3 = 1)$$
$$\wedge \quad (Y_2 = 1 \implies Y_3 = 1)$$



Pipatsrisawat and Darwiche, "New Compilation Languages Based on Structured Decomposability.", AAAI, 2008

knowledge compilation

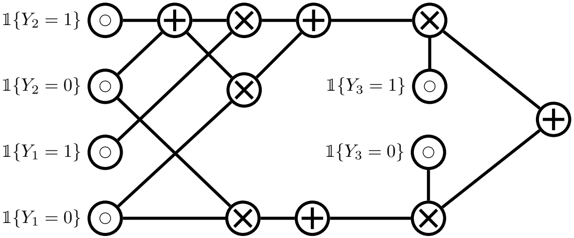
$$\mathsf{K}: (Y_1 = 1 \implies Y_3 = 1)$$
$$\wedge \quad (Y_2 = 1 \implies Y_3 = 1)$$



Pipatsrisawat and Darwiche, "New Compilation Languages Based on Structured Decomposability.", AAAI, 2008

knowledge compilation

$$\mathsf{K} : \ (Y_1 = 1 \implies Y_3 = 1)$$
$$\land \ (Y_2 = 1 \implies Y_3 = 1)$$



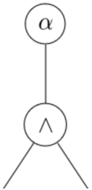
Pipatsrisawat and Darwiche, "New Compilation Languages Based on Structured Decomposability.", AAAI, 2008

main cirkit / notebooks / logic-circuits.ipynb

Preview Code Blame 1183 lines (1183 loc) · 81.5 KB Raw

Compiling a propositional formula using `cirkit`

As we said before, a propositional formula can be easily represented as a circuit with a tree-structure. For instance, α presented before can be represented as the following tree:



check the notebook

how

to enforce constraints?

$$\max p_\theta(\mathsf{K}_i)$$

maximise the probability of the constraint to hold!

Xu et al., "A Semantic Loss Function for Deep Learning with Symbolic Knowledge", Proceedings of the 35th International Conference on Machine Learning (ICML), 2018

how

to enforce constraints?

$$\min \mathcal{L}(\mathsf{K}_i, p_\theta) = \min -\log \sum_{\mathbf{z} \models \mathsf{K}_i} \prod_{j:\mathbf{z} \models z_{f_j}} p_\theta(z_{f_j}) \prod_{j:\mathbf{z} \models \neg z_{f_j}} (1 - p_\theta(z_{f_j}))$$

minimize the semantic loss

Xu et al., "A Semantic Loss Function for Deep Learning with Symbolic Knowledge", Proceedings of the 35th International Conference on Machine Learning (ICML), 2018

WMC

computing the probability of logical formulas

$$p_\theta(\mathsf{K}(\mathbf{z})) = \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\mathbb{1}\{\mathbf{z} \models \mathsf{K}\}]$$

computing the probability of K

Xu et al., "A Semantic Loss Function for Deep Learning with Symbolic Knowledge", Proceedings of the 35th International Conference on Machine Learning (ICML), 2018

WMC

computing the probability of logical formulas

$$\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\mathbb{1}\{\mathbf{z} \models \mathsf{K}\}] = \sum_{\mathbf{z}} p(\mathbf{z})\mathbb{1}\{\mathbf{z} \models \mathsf{K}\} = \sum_{\mathbf{z}\models\mathsf{K}} p(\mathbf{z})$$

computing the ***weighted model count*** *(WMC) of* K

Xu et al., "A Semantic Loss Function for Deep Learning with Symbolic Knowledge", Proceedings of the 35th International Conference on Machine Learning (ICML), 2018

WMC

computing the probability of logical formulas

$$\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\mathbb{1}\{\mathbf{z} \models \mathsf{K}\}] = \sum_{\mathbf{z}\models\mathsf{K}} \prod_{i:\mathbf{z}\models z_i} p(z_i) \prod_{i:\mathbf{z}\models\neg z_i} (1 - p(z_i))$$

assuming independence of $\mathbf{z}$ (but be careful!)[1]

[1] *van Krieken et al., "On the Independence Assumption in Neurosymbolic Learning", 2024*

Xu et al., "A Semantic Loss Function for Deep Learning with Symbolic Knowledge", Proceedings of the 35th International Conference on Machine Learning (ICML), 2018

WMC

computing the probability of logical formulas

$$\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\mathbb{1}\{\mathbf{z} \models \mathsf{K}\}] = \sum_{\mathbf{z}\models \mathsf{K}} \prod_{i:\mathbf{z}\models z_i} p(z_i) \prod_{i:\mathbf{z}\models \neg z_i} (1 - p(z_i))$$

computing WMC is #P-hard in general : (

Xu et al., "A Semantic Loss Function for Deep Learning with Symbolic Knowledge", Proceedings of the 35th International Conference on Machine Learning (ICML), 2018

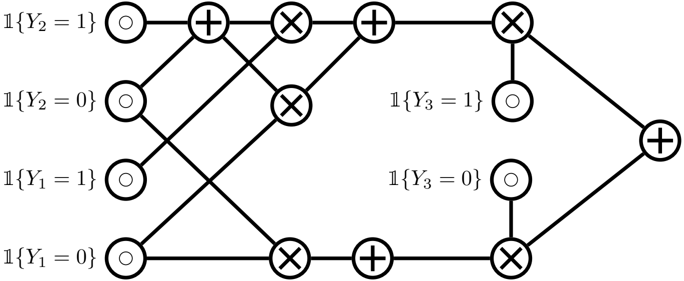
Goal

Can we encode K
to yield a tractable WMC?

Goal

Can we encode K
to yield a tractable WMC?
yes, as a circuit!

tractable WMC



exactly compute WMC in time $O(|c|)$

Vergari *et al., "A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference", NeurIPS, 2021*

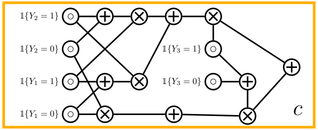
SL recipe

$$\mathsf{K}: (Y_1 = 1 \implies Y_3 = 1)$$
$$\land \quad (Y_2 = 1 \implies Y_3 = 1)$$

1) Take a
logical constraint

SL recipe

K : $(Y_1 = 1 \implies Y_3 = 1)$
$\wedge \ (Y_2 = 1 \implies Y_3 = 1)$



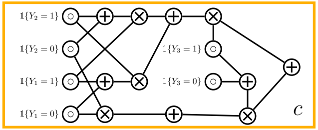
1) Take a logical constraint

2) Compile it into a constraint circuit

SL recipe

$$\mathsf{K} : (Y_1 = 1 \implies Y_3 = 1)$$
$$\wedge \quad (Y_2 = 1 \implies Y_3 = 1)$$

1) Take a logical constraint



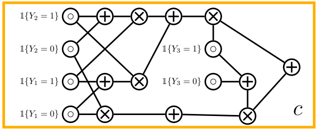
2) Compile it into a constraint circuit

$$-\log \mathsf{WMC}(\mathsf{K}_i, p_\theta)$$

3) minimize the semantic loss

SL recipe

$\mathsf{K} : (Y_1 = 1 \implies Y_3 = 1)$
$\wedge \quad (Y_2 = 1 \implies Y_3 = 1)$



$-\log \mathrm{WMC}(\mathsf{K}_i, p_\theta)$

1) Take a logical constraint

2) Compile it into a constraint circuit

3) minimize the semantic loss

4) train end-to-end by sgd!

cirkit / notebooks / **semantic-loss.ipynb**

Preview | Code | Blame — 625 lines (625 loc) · 62.2 KB — Raw

Semantic loss using `cirkit`

So far, we used our logic circuit in a *traditional* setting, by running queries related to its logical semantics. However, logic circuits also enable neuro-symbolic methods! In this notebook, we implement the **semantic loss** (Xu et al, 2018), which relies on a logic circuit to implement contraints on the predictions of a neural network, using an SDD.

Imagine we want to enforce constraints on the output $\hat{y}$ of a neural network and we want to model those contraints as a logic formula F by considering each element of $\hat{y}$ as a literal in F. We can compile the formula F into a logic circuit c and we can efficiently perform a bunch of operations on it (see logic circuits), but can we actually use the values of $\hat{y}$ as inputs to c? Or in other words, can we quantify if the input y is a model for c ($\mathsf{y} \models \mathsf{c}$)?

https://github.com/n28div/cirkit/blob/main/notebooks/semantic-loss.ipynb

nice...

...but!

assuming facts to be independent...

nice...

...but!

assuming facts to be independent...

no guarantees to satisfy constraints at test time...

nice...

...but!

assuming facts to be independent...

no guarantees to satisfy constraints at test time...

WMC

on the independence assumption

$$K : \neg \mathbf{r} \vee \neg \mathbf{g}$$

a neural net should not output that a traffic light is both ***red*** *and* ***green***

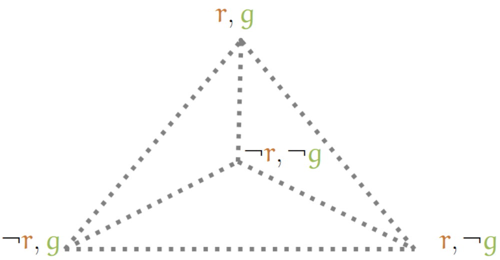
van Krieken et al., "On the Independence Assumption in Neurosymbolic Learning", ICML, 2024

WMC

on the independence assumption

$$K : \neg \mathbf{r} \vee \neg \mathbf{g}$$

a neural net should not output that a traffic light is both ***red*** *and* ***green***



van Krieken et al., "On the Independence Assumption in Neurosymbolic Learning", ICML, 2024

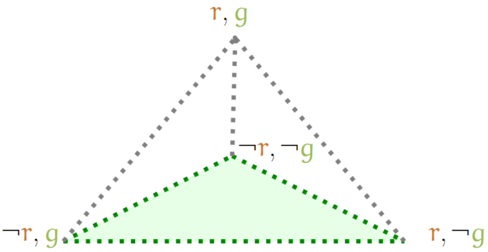
WMC

on the independence assumption

$$K : \neg r \vee \neg g$$

*a neural net should not output that a traffic light is both **red** and **green***

only some probability assignments should be non-zero (lower triangle)



van Krieken et al., "On the Independence Assumption in Neurosymbolic Learning", ICML, 2024

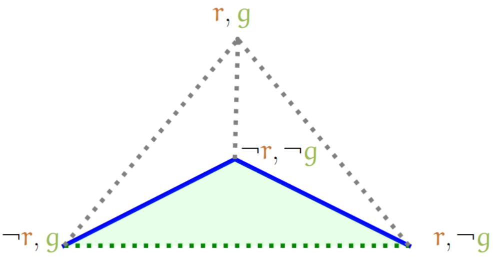
WMC

on the independence assumption

$$K : \neg \mathbf{r} \vee \neg \mathbf{g}$$

a neural net should not output that a traffic light is both ***red*** *and* ***green***

but assuming $p(\mathbf{r}, \mathbf{g}) = p(\mathbf{r})p(\mathbf{g})$ *restricts this even further (only blue lines)*



van Krieken et al., "On the Independence Assumption in Neurosymbolic Learning", ICML, 2024

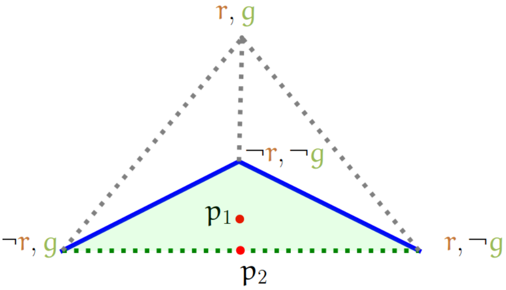
WMC

on the independence assumption

$$K : \neg \mathbf{r} \lor \neg \mathbf{g}$$

*a neural net should not output that a traffic light is both **red** and **green***

but assuming $p(\mathbf{r}, \mathbf{g}) = p(\mathbf{r})p(\mathbf{g})$ *restricts this even further (only blue lines)*



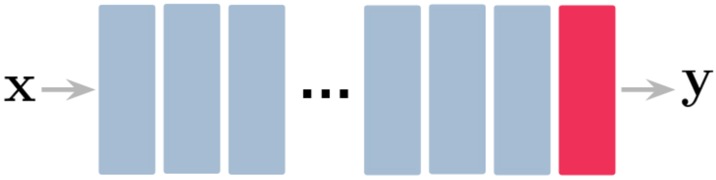
nice...

...but!

assuming facts to be independent...

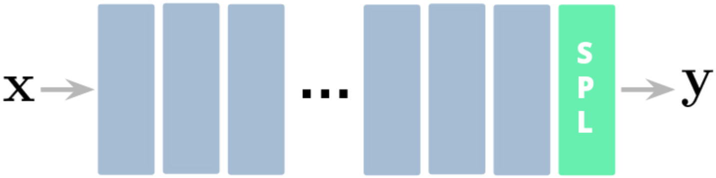
no guarantees to satisfy constraints at test time...

how to



make any neural network architecture...

how to



...guarantee all predictions to conform to constraints?

Semantic Probabilistic Layers for Neuro-Symbolic Learning

Kareem Ahmed
CS Department
UCLA
ahmedk@cs.ucla.edu

Stefano Teso
CIMeC and DISI
University of Trento
stefano.teso@unitn.it

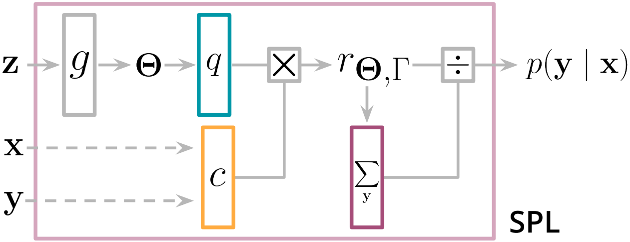
Kai-Wei Chang
CS Department
UCLA
kwchang@cs.ucla.edu

Guy Van den Broeck
CS Department
UCLA
guyvdb@cs.ucla.edu

Antonio Vergari
School of Informatics
University of Edinburgh
avergari@ed.ac.uk

***enforce constraints in neural networks* at NeurIPS 2022**

SPL

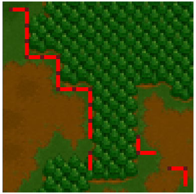


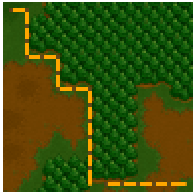
$$p(\mathbf{y} \mid \mathbf{x}) = \boldsymbol{q}_{\boldsymbol{\Theta}}(\mathbf{y} \mid g(\mathbf{z})) \cdot \boldsymbol{c}_{\mathsf{K}}(\mathbf{x}, \mathbf{y}) / \mathcal{Z}(\mathbf{x})$$

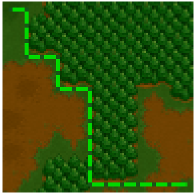
$$\mathcal{Z}(\mathbf{x}) = \sum_{\mathbf{y}} q_{\boldsymbol{\Theta}}(\mathbf{y} \mid \mathbf{x}) \cdot c_{\mathsf{K}}(\mathbf{x}, \mathbf{y})$$

SPL


Ground Truth


ResNet-18


Semantic Loss


circuits

predictions guarantee a logical constraint 100% of the time!

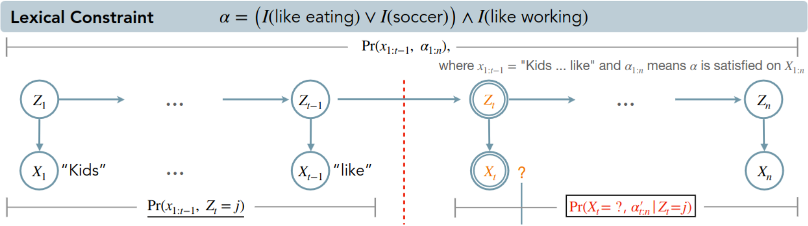
SPL

(and variants)

everywhere

Tractable Control for Autoregressive Language Generation

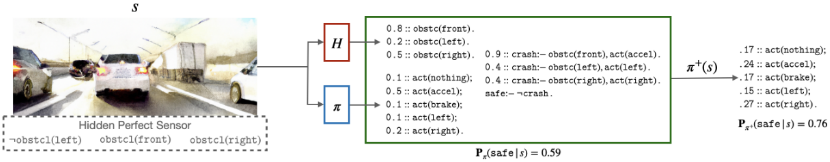
Honghua Zhang * [1] **Meihua Dang** * [1] **Nanyun Peng** [1] **Guy Van den Broeck** [1]



constrained text generation with LLMs (ICML 2023)

Safe Reinforcement Learning via Probabilistic Logic Shields

Wen-Chi Yang[1] , Giuseppe Marra[1] , Gavin Rens and Luc De Raedt[1,2]



reliable reinforcement learning (AAAI 23)

How to Turn Your Knowledge Graph Embeddings into Generative Models

Lorenzo Loconte
University of Edinburgh, UK
l.loconte@sms.ed.ac.uk

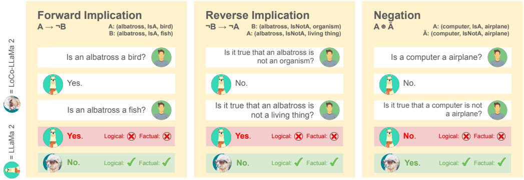
Nicola Di Mauro
University of Bari, Italy
nicola.dimauro@uniba.it

Robert Peharz
TU Graz, Austria
robert.peharz@tugraz.at

Antonio Vergari
University of Edinburgh, UK
avergari@ed.ac.uk

enforce constraints in knowledge graph embeddings
oral at NeurIPS 2023

Logically Consistent Language Models via Neuro-Symbolic Integration



improving logical (self-)consistency in LLMs at ICLR 2025

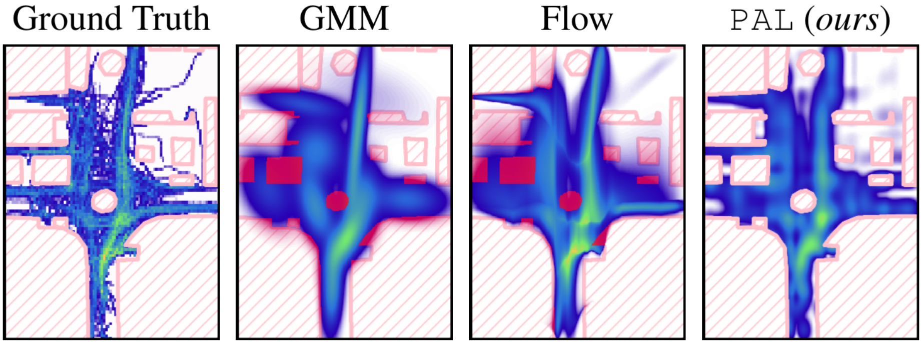
open problems

I constraints over continuous variables

II scaling to H U G E constraints

III learn (partial) constraints

IV revise constraints (continual learning)



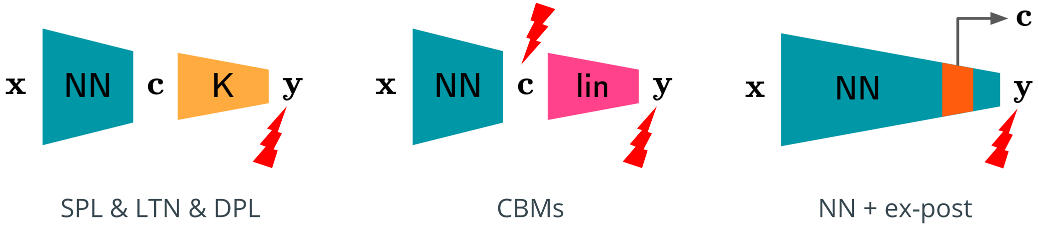
extending it to SMT constraints

A Probabilistic Neuro-symbolic Layer for Algebraic Constraint Satisfaction

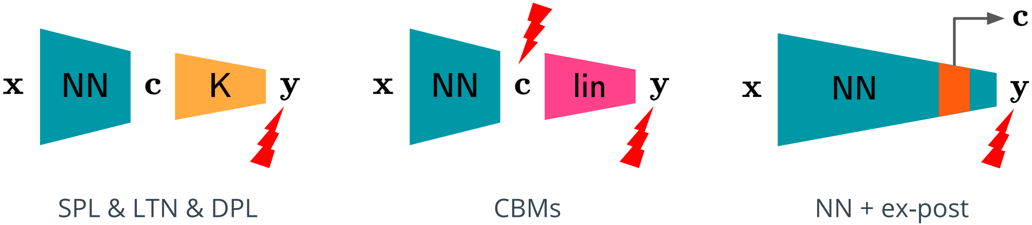
Leander Kurscheidt[1] Paolo Morettin[2] Roberto Sebastiani[2] Andrea Passerini[2] Antonio Vergari[1]

[1]School of Informatics, University of Edinburgh, UK
[2]DiSI, University of Trento, Italy

extending it to SMT constraints



NeSy models are concept bottlenecks

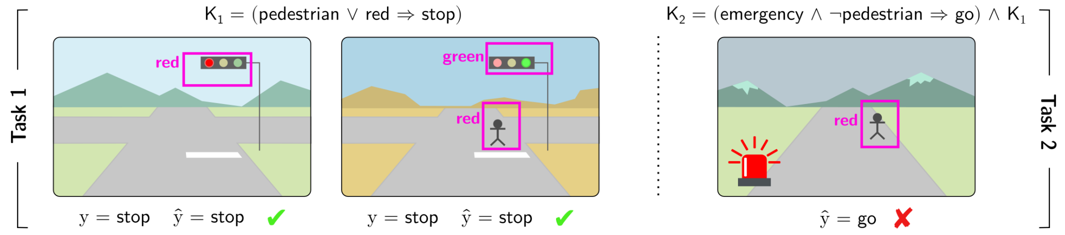


NeSy models can suffer from reasoning shortcuts!

Task	Example Data	Knowledge K	Example RS	Impact
MNIST math	$2 \cdot$  $+$  $= 6$  $+$  $= 7$	Equations must hold.	 $\to 2$  $\to 4$  $\to 3$	 $+$  $= 5$

NeSy models can suffer from reasoning shortcuts!

Bortolotti et al., "A Benchmark Suite for Systematically Evaluating Reasoning Shortcuts", NeurIPS Benchmark track, 2024



how to detect and mitigate them

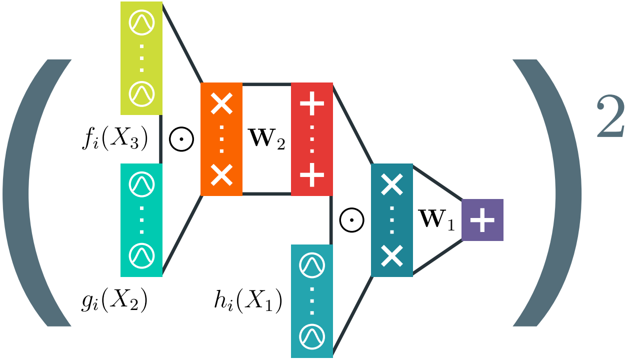
Marconato et al., "Not all neuro-symbolic concepts are created equal: Analysis and mitigation of reasoning shortcuts", NeurIPS, 2023

Bortolotti et al., "A Benchmark Suite for Systematically Evaluating Reasoning Shortcuts", NeurIPS Benchmark track, 2024

cirkit

learning & reasoning with circuits in pytorch

`github.com/april-tools/cirkit`



questions?