Provably reliable Deep learning by design

antonio vergari (he/him)

@tetraduzione

joint work with Kareem Ahmed, Andreas Grivas, Lorenzo Loconte, Stefano Teso, Kai-Wei Chang, Guy Van den Broeck, Nicola di Mauro, Robert Peharz, Adam Lopez

27th Oct 2023 - **Workshop on Safe and Robust Machine Learning**

april

***a**pril is*
***p**robably a*
***r**ecursive*
***i**dentifier of a*
***l**ab*

april

***a**utonomous &*
***p**rovably*
***r**eliable*
***i**ntelligent*
***l**earners*

deep learning is differentiable lego

just stacking blocks can be unrealiable though...

especially when dealing with hard constraints

part I how to satisfy wanted constraints in NNs by design

part II how the design of NNs implicitly poses unwanted constraints

part I how to satisfy wanted constraints in NNs by design

part II how the design of NNs implicitly poses unwanted constraints

part I how to satisfy wanted constraints in NNs by design

Why?

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

Why?

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

integrate ***hard*** **(logical)** constraints

Why?

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

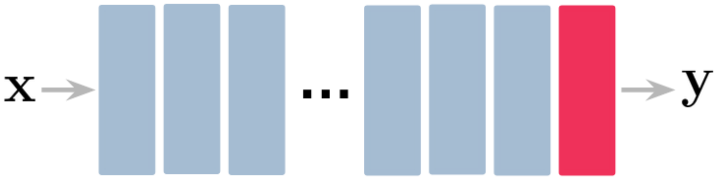
guarantee that predictions ***always satisfy*** constraints

Why?

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

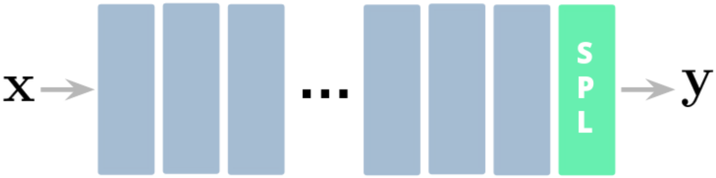
fast and ***exact*** gradients

Why?



make any neural network architecture...

Why?



...guarantee all predictions to conform to constraints

When?

Ground Truth

e.g. predict shortest path in a map

When?

Ground Truth

given $\mathbf{x}$ *// e.g. a tile map*

nesy structured output prediction (SOP) tasks

Vlastelica et al., "Differentiation of blackbox combinatorial solvers", ICLR, 2020

When?

Ground Truth

given $\mathbf{x}$ *// e.g. a tile map*

find $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}} p_\theta(\mathbf{y} \mid \mathbf{x})$ *// e.g. a configurations of edges in a grid*

nesy structured output prediction (SOP) tasks

Vlastelica et al., "Differentiation of blackbox combinatorial solvers", ICLR, 2020

When?

Ground Truth

given $\mathbf{x}$ *// e.g. a tile map*

find $\mathbf{y}^* = \mathrm{argmax}_{\mathbf{y}}\, p_\theta(\mathbf{y} \mid \mathbf{x})$ *// e.g. a configurations of edges in a grid*

s.t. $\mathbf{y} \models \mathsf{K}$ *// e.g., that form a valid path*

nesy structured output prediction (SOP) tasks

Vlastelica et al., "Differentiation of blackbox combinatorial solvers", ICLR, 2020

When?

Ground Truth

given $\mathbf{x}$ *// e.g. a tile map*

find $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}} p_\theta(\mathbf{y} \mid \mathbf{x})$ *// e.g. a configurations of edges in a grid*

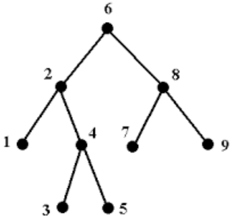
s.t. $\mathbf{y} \models \mathsf{K}$ *// e.g., that form a valid path*

// for a 12×12 grid, 2^{144} states but only 10^{10} valid ones!

nesy structured output prediction (SOP) tasks

Vlastelica et al., "Differentiation of blackbox combinatorial solvers", ICLR, 2020

When?



given $\mathbf{x}$ *// e.g. a feature map*

find $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}} p_\theta(\mathbf{y} \mid \mathbf{x})$ *// e.g. labels of classes*

s.t. $\mathbf{y} \models \mathsf{K}$ *// e.g., constraints over superclasses*

$$\mathsf{K} : (Y_{\mathsf{cat}} \implies Y_{\mathsf{animal}}) \wedge (Y_{\mathsf{dog}} \implies Y_{\mathsf{animal}})$$

hierarchical multi-label classification

Giunchiglia and Lukasiewicz, "Coherent hierarchical multi-label classification networks", NeurIPS, 2020

When?

given $\mathbf{x}$ *// e.g. a user preference over $K - N$ sushi types*

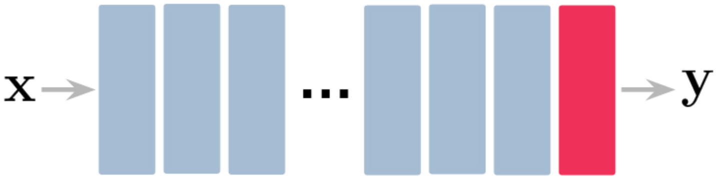
find $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}} p_\theta(\mathbf{y} \mid \mathbf{x})$ *// e.g. prefs over N more types*

s.t. $\mathbf{y} \models \mathsf{K}$ *// e.g., output valid rankings*

user preference learning

Choi, Van den Broeck, and Darwiche, "Tractable learning for structured probability spaces: A case study in learning preference distributions", IJCAI, 2015

e.g.,

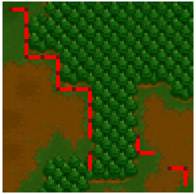


sigmoid linear layers

$$p(\mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^{N} p(y_i \mid \mathbf{x})$$

When?

Ground Truth



ResNet-18

neural nets struggle to satisfy validity constraints!

the issues!

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

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

II) How to integrate logic and probabilities in a single neural layer

$\Longrightarrow$ ***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 matching, paths, hierarchies, ...*

II) How to integrate logic and probabilities in a single neural layer

$\Longrightarrow$ ***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 matching, paths, hierarchies, ...*

II) How to integrate logic and probabilities in a single neural layer

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

III) Logical constraints are piecewise constant functions!

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

Constraint losses

$$\mathcal{L}(\theta; \mathbf{x}, \mathbf{y}) + \lambda \mathcal{L}_{\mathsf{K}}(\mathbf{x}, \mathbf{y})$$

losses improve consistency during training...

Xu et al., "A Semantic Loss Function for Deep Learning with Symbolic Knowledge", , 2018

Constraint losses

$$\mathcal{L}(\theta; \mathbf{x}, \mathbf{y}) + \lambda \mathcal{L}_{\mathsf{K}}(\mathbf{x}, \mathbf{y})$$

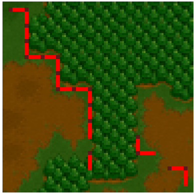
losses improve consistency during training...

e.g., the ***semantic loss***: $\mathcal{L}_{\mathsf{SL}} := -\log \sum_{\mathbf{y} \models \mathsf{K}} \prod_i p(Y_i \mid \mathbf{x})$

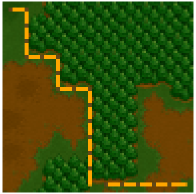
Xu et al., "A Semantic Loss Function for Deep Learning with Symbolic Knowledge", , 2018

Constraint losses

Ground Truth



ResNet-18



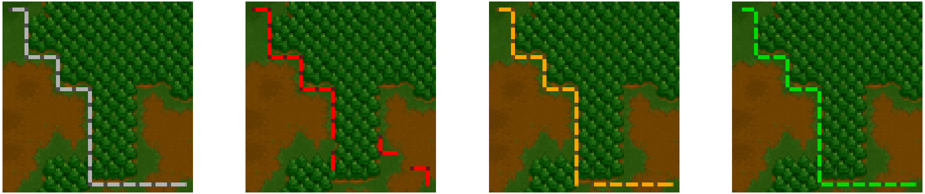
Semantic Loss

...but cannot guarantee consistency at test time!

what?

	Losses			Layers				
Desideratum	DL2 [29]	SL [80]	NeSyEnt [3]	FIL	EBM [43]	MultiplexNet [38]	CCN [33]	SPL *(ours)*
(D1) Probabilistic	✗	✓	✓	✓	✗	✓	✗	✓
(D2) Expressive	✗	✗	✗	✗	✓	✗	✗	✓
(D3) Consistent	✗	✗	✗	✗	✗	✓	✓	✓
(D4) General	✓	✓	✓	✗	✓	✓	✗	✓
(D5) Modular	✓	✓	✓	✓	✓	✓	✓	✓
(D6) Efficient	✓	✓	✓	✓	✗	✗	✓	✓

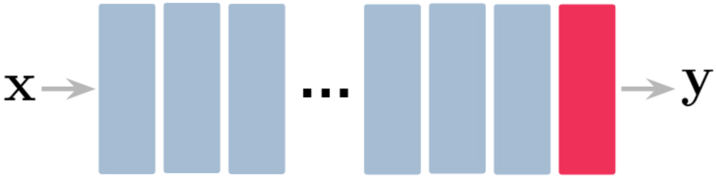
SPL



Ground Truth ResNet-18 Semantic Loss SPL (ours)

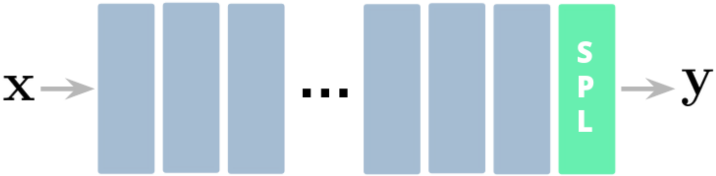
you can predict valid paths 100% of the time!

How?



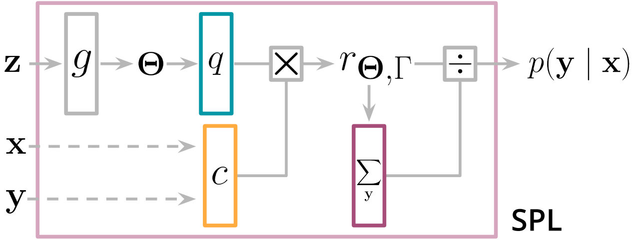
take an unreliable neural network architecture...

How?



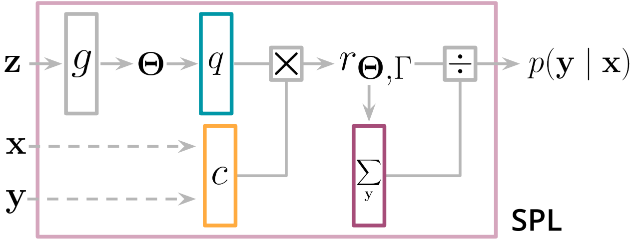
......and replace the last layer with a semantic probabilistic layer

SPL



$\mathbf{z}$
g
$\boldsymbol{\Theta}$
q
$r_{\boldsymbol{\Theta},\Gamma}$
$p(\mathbf{y} \mid \mathbf{x})$
$\mathbf{x}$
$\mathbf{y}$
c
$\sum_{\mathbf{y}}$
SPL

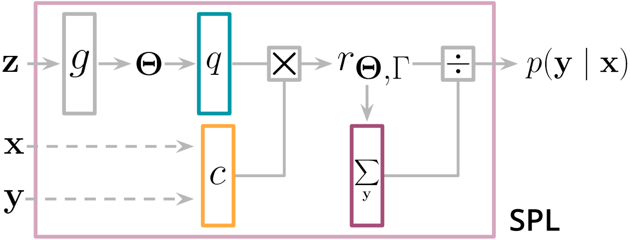
SPL



$$p(\mathbf{y} \mid \mathbf{x}) = q_{\boldsymbol{\Theta}}(\mathbf{y} \mid g(\mathbf{z}))$$

$q_{\boldsymbol{\Theta}}(\mathbf{y} \mid g(\mathbf{z}))$ is an expressive distribution over labels

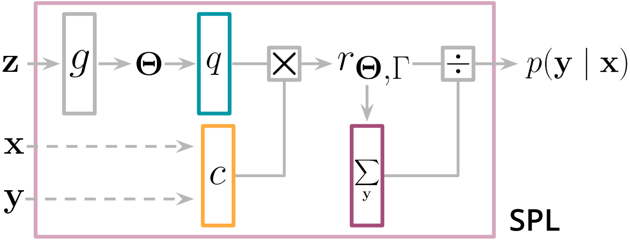
SPL



$$p(\mathbf{y} \mid \mathbf{x}) = q_{\boldsymbol{\Theta}}(\mathbf{y} \mid g(\mathbf{z})) \cdot c_{\mathsf{K}}(\mathbf{x}, \mathbf{y})$$

$c_{\mathsf{K}}(\mathbf{x}, \mathbf{y})$ encodes the constraint $\mathbb{1}\{\mathbf{x}, \mathbf{y} \models \mathsf{K}\}$

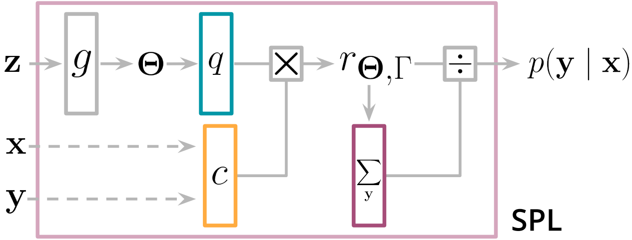
SPL



$$p(\mathbf{y} \mid \mathbf{x}) = q_{\Theta}(\mathbf{y} \mid g(\mathbf{z})) \cdot c_{\mathsf{K}}(\mathbf{x}, \mathbf{y})$$

a product of experts : (

SPL



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

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

Goal

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

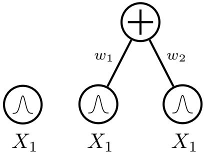
Probabilistic Circuits (PCs)

A grammar for tractable computational graphs

I. *A simple tractable function 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*

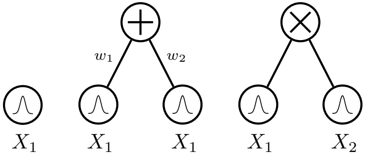
Probabilistic Circuits (PCs)

A grammar for tractable computational graphs

I. *A simple tractable function is a circuit*

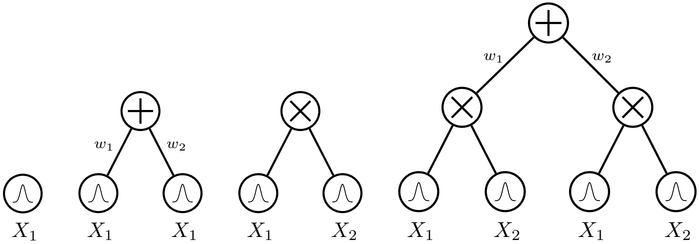
II. *A weighted combination of circuits is a circuit*

III. *A product of circuits is a circuit*



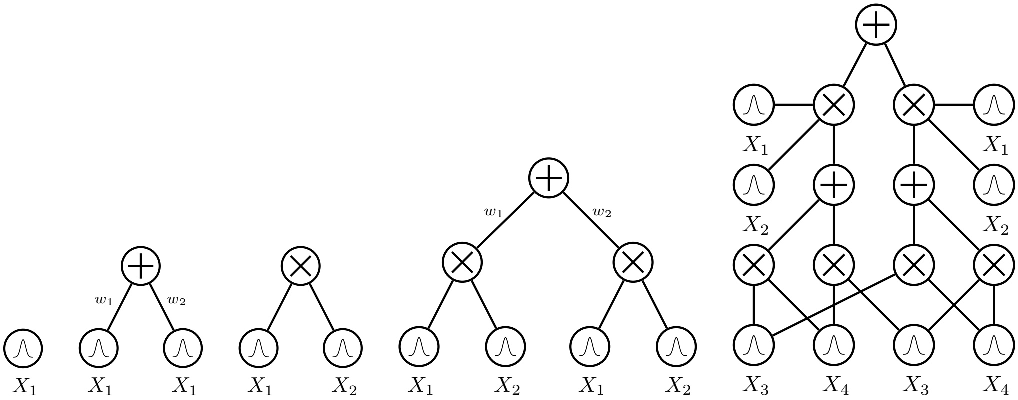
Probabilistic Circuits (PCs)

A grammar for tractable computational graphs



Probabilistic Circuits (PCs)

A grammar for tractable computational graphs



...why PCs?

1. A grammar for tractable models

One formalism to represent many models. *#HMMs #Trees #XGBoost*, ...

2. Expressiveness

Competitive with intractable models, VAEs, Flow...*#hierachical #mixtures #polynomials*

...why PCs?

1. A grammar for tractable models

One formalism to represent many models. *#HMMs #Trees #XGBoost*, ...

2. Expressiveness

Competitive with intractable models, VAEs, Flow...*#hierachical #mixtures #polynomials*

3. Tractability == Structural Properties!!!

Exact computations of reasoning tasks are certified by guaranteeing certain structural properties. *#marginals #expectations #MAP*, *#product* ...

Structural properties

smoothness

decomposability

determinism

compatibility

Vergari *et al., "A Compositional Atlas of Tractable Circuit Operations: From Simple Transformations to Complex Information-Theoretic Queries", NeurIPS, 2021*

Structural properties

property A

property B

property C

property D

Vergari *et al., "A Compositional Atlas of Tractable Circuit Operations: From Simple Transformations to Complex Information-Theoretic Queries", NeurIPS, 2021*

Structural properties

property A

property B

property C

property D

tractable computation of ***arbitrary integrals***

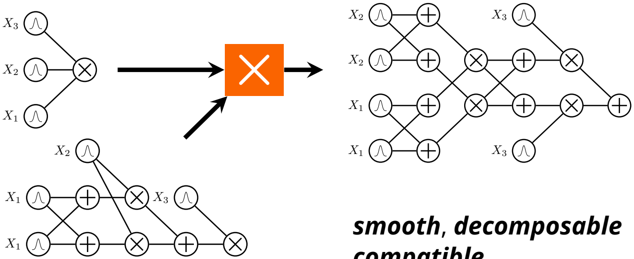
$$p(\mathbf{y}) = \int_{\mathsf{val}(\mathbf{Z})} p(\mathbf{z}, \mathbf{y})\, d\mathbf{Z}, \quad \forall \mathbf{Y} \subseteq \mathbf{X}, \quad \mathbf{Z} = \mathbf{X} \setminus \mathbf{Y}$$

$\implies$ ***sufficient*** *and* ***necessary*** *conditions for a single feedforward evaluation*

$\implies$ *tractable partition function*

__Vergari__ et al., "A Compositional Atlas of Tractable Circuit Operations: From Simple Transformations to Complex Information-Theoretic Queries", NeurIPS, 2021

Tractable products

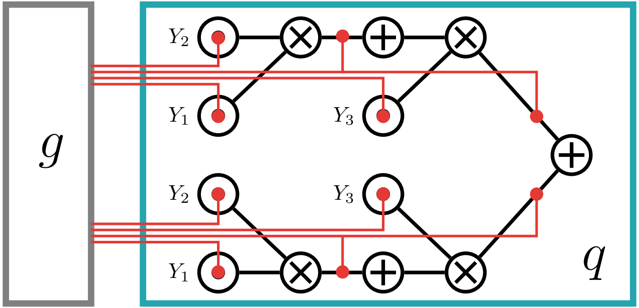


smooth, ***decomposable***
compatible

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

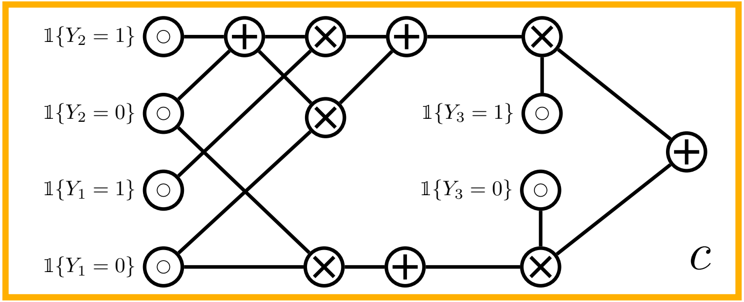
__Vergari__ et al., "A Compositional Atlas of Tractable Circuit Operations: From Simple Transformations to Complex Information-Theoretic Queries", NeurIPS, 2021

SPL



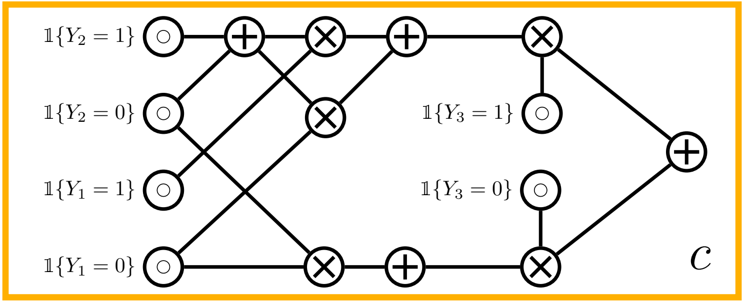
a conditional circuit $\boldsymbol{q}(\mathbf{y}; \boldsymbol{\Theta} = g(\mathbf{z}))$

SPL



and a logical circuit $c(\mathrm{y}, \mathrm{x})$ **encoding** K

SPL



compiling logical formulas into circuits

Knowledge compilation

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

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

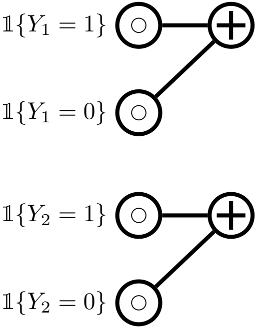
$\mathbb{1}\{Y_1 = 0\}$ $\odot$ $\mathbb{1}\{Y_1 = 1\}$ $\odot$ $\mathbb{1}\{Y_2 = 0\}$ $\odot$

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

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

Knowledge compilation

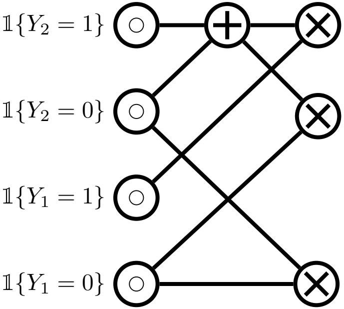
$$\mathsf{K} : \ (Y_1 = 1 \implies Y_3 = 1)$$
$$\wedge \ \ (Y_2 = 1 \implies Y_3 = 1)$$



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

Knowledge compilation

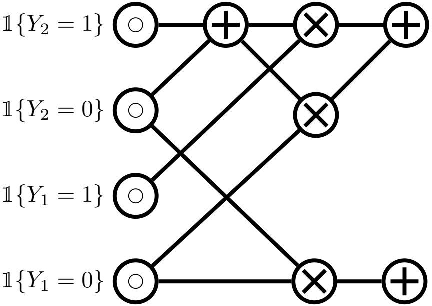
$$\begin{aligned} \mathsf{K}: \ & (Y_1 = 1 \implies Y_3 = 1) \\ \wedge \ & (Y_2 = 1 \implies Y_3 = 1) \end{aligned}$$



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

Knowledge compilation

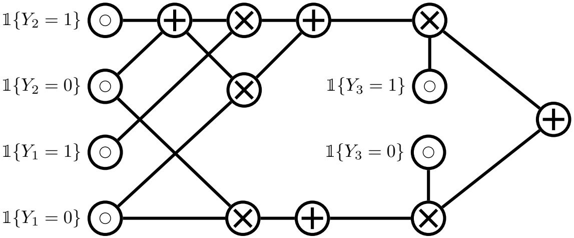
$$\mathsf{K}: \ (Y_1 = 1 \implies Y_3 = 1)$$
$$\wedge \ (Y_2 = 1 \implies Y_3 = 1)$$



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

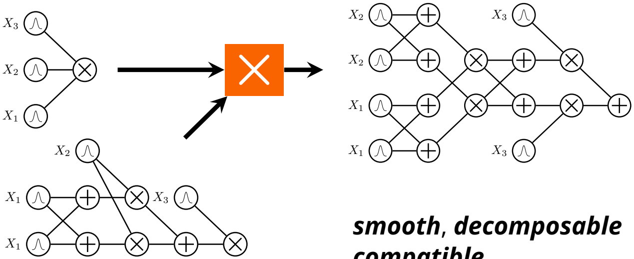
Knowledge compilation

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



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

Tractable products



smooth*, *decomposable
compatible

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

__Vergari__ et al., "A Compositional Atlas of Tractable Circuit Operations: From Simple Transformations to Complex Information-Theoretic Queries", NeurIPS, 2021

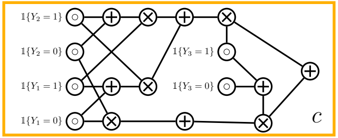
SPL recipe

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

1) Take a
logical constraint

SPL recipe

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

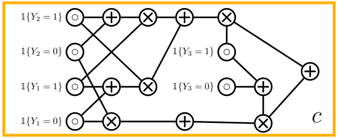


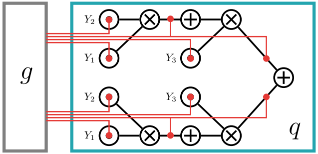
1) Take a logical constraint

2) Compile it into a constraint circuit

SPL recipe

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





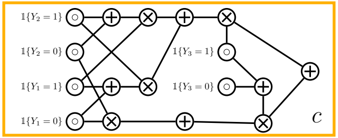
1) Take a logical constraint

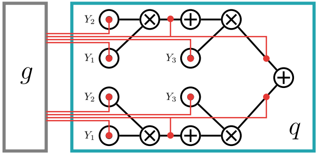
2) Compile it into a constraint circuit

3) Multiply it by a circuit distribution

SPL recipe

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





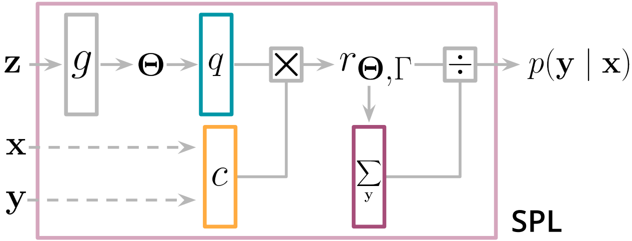
1) Take a logical constraint

2) Compile it into a constraint circuit

3) Multiply it by a circuit distribution

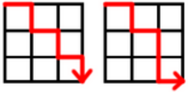
4) train end-to-end by sgd!

Experiments



how good are SPLs?

Experiments

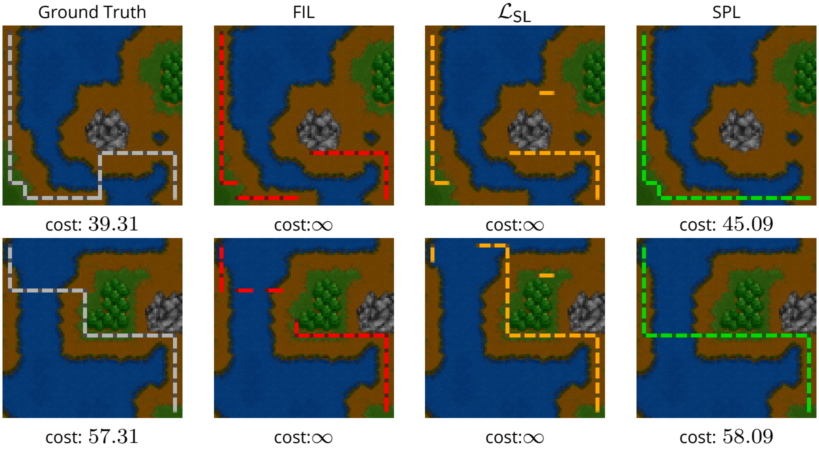


Architecture	Simple Path			Preference Learning		
	Exact	Hamming	Consistent	Exact	Hamming	Consistent
MLP+FIL	5.6	85.9	7.0	1.0	**75.8**	2.7
MLP+$\mathcal{L}_{SL}$	28.5	83.1	75.2	15.0	72.4	69.8
MLP+NeSyEnt	30.1	83.0	91.6	18.2	71.5	96.0
MLP+SPL	**37.6**	**88.5**	**100.0**	**20.8**	72.4	**100.0**

Experiments

Architecture	Exact	Hamming	Consistent
ResNet-18+FIL	55.0	**97.7**	56.9
ResNet-18+$\mathcal{L}_{\mathrm{SL}}$	59.4	**97.7**	61.2
ResNet-18+SPL	**78.2**	96.3	**100.0**

Experiments



Experiments

Dataset	Exact Match	
	HMCNN	MLP+SPL
CellCycle	3.05 ± 0.11	$\mathbf{3.79 \pm 0.18}$
Derisi	1.39 ± 0.47	$\mathbf{2.28 \pm 0.23}$
Eisen	5.40 ± 0.15	$\mathbf{6.18 \pm 0.33}$
Expr	4.20 ± 0.21	$\mathbf{5.54 \pm 0.36}$
Gasch1	3.48 ± 0.96	$\mathbf{4.65 \pm 0.30}$
Gasch2	3.11 ± 0.08	$\mathbf{3.95 \pm 0.28}$
Seq	5.24 ± 0.27	$\mathbf{7.98 \pm 0.28}$
Spo	$\mathbf{1.97 \pm 0.06}$	1.92 ± 0.11
Diatoms	48.21 ± 0.57	$\mathbf{58.71 \pm 0.68}$
Enron	5.97 ± 0.56	$\mathbf{8.18 \pm 0.68}$
ImCLEF07A	79.75 ± 0.38	$\mathbf{86.08 \pm 0.45}$
ImCLEF07D	76.47 ± 0.35	$\mathbf{81.06 \pm 0.68}$

How to Turn Your Knowledge Graph Embeddings into Generative Models via Probabilistic Circuits

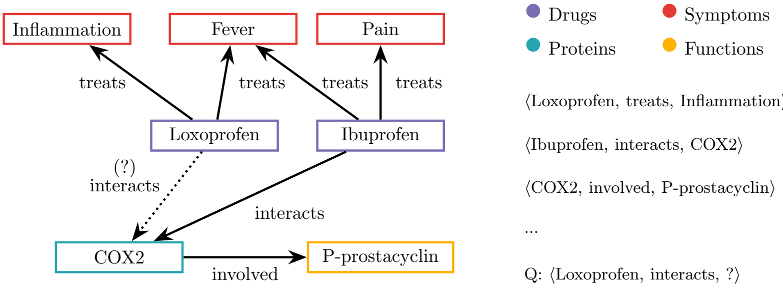
Lorenzo Loconte
University of Edinburgh, UK
l.loconte@sms.ed.ac.uk

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

SPL meets knowledge graph embedding models



⟨Loxoprofen, treats, Inflammation⟩

⟨Ibuprofen, interacts, COX2⟩

⟨COX2, involved, P-prostacyclin⟩

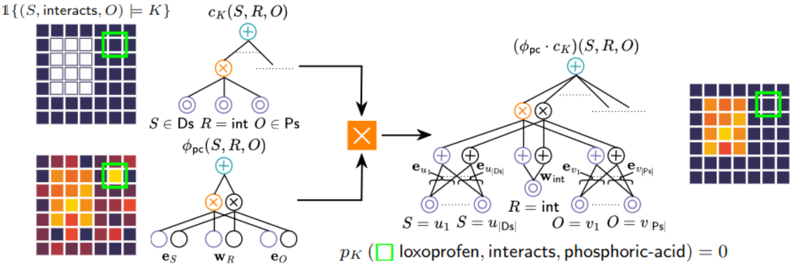
...

Q: ⟨Loxoprofen, interacts, ?⟩

A: ⟨Loxoprofen, interacts, **phosphoric-acid**⟩ !!!

neural link predictors can violate domain constraints

SPL+KGE

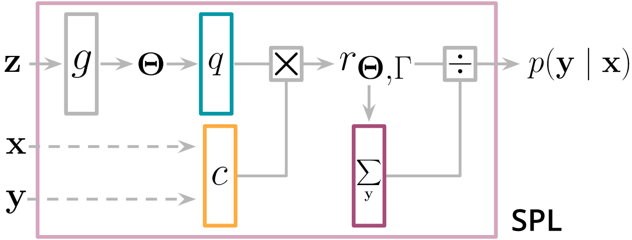


$\mathbb{1}\{(S, \text{interacts}, O) \models K\}$
$c_K(S, R, O)$
$S \in \text{Ds}$ $R = \text{int}$ $O \in \text{Ps}$
$\phi_{\text{pc}}(S, R, O)$
$\mathbf{e}_S$ $\mathbf{w}_R$ $\mathbf{e}_O$
$(\phi_{\text{pc}} \cdot c_K)(S, R, O)$
$\mathbf{e}_{u_1}$ $\mathbf{e}_{u_{|\text{Ds}|}}$ $\mathbf{w}_{\text{int}}$ $\mathbf{e}_{v_1}$ $\mathbf{e}_{v_{|\text{Ps}|}}$
$S = u_1$ $S = u_{|\text{Ds}|}$ $R = \text{int}$ $O = v_1$ $O = v_{|\text{Ps}|}$
p_K (loxoprofen, interacts, phosphoric-acid) $= 0$


MACHINE LEARNERS
COMPLEX REASONING

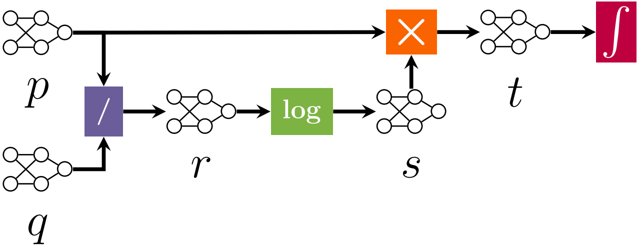

APPROXIMATIONS
imgflip.com

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



efficient and reliable reasoning over constraints

$$\int p(\mathbf{x}) \times \log \left(p(\mathbf{x}) \,/\, q(\mathbf{x}) \right) \, d\mathbf{X}$$



build a LEGO-like query calculus!

Vergari *et al., "A Compositional Atlas of Tractable Circuit Operations: From Simple Transformations to Complex Information-Theoretic Queries", NeurIPS, 2021*

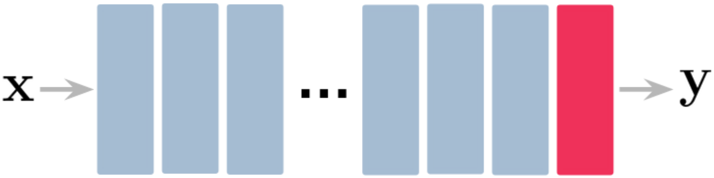
part I how to satisfy wanted constraints in NNs by design

part II how the design of NNs implicitly poses unwanted constraints

Taming the Sigmoid Bottleneck: Provably Argmaxable Sparse Multi-Label Classification

Andreas Grivas[1], Antonio Vergari†[2], Adam Lopez†[1]

[1] Institute for Language, Cognition, and Computation
[2] Institute for Adaptive and Neural Computation
School of Informatics, University of Edinburgh
{agrivas, avergari, alopez}@ed.ac.uk



sigmoid linear layers

$$p(\mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^{n} p(y_i \mid \mathbf{x})$$

low-rank classifiers

labels (n) $>>$ embedding size (d)

e.g.,

clinically annotated text

$n \approx 9000$
$d \approx 200 - 500$

large-scale biomedical question answering

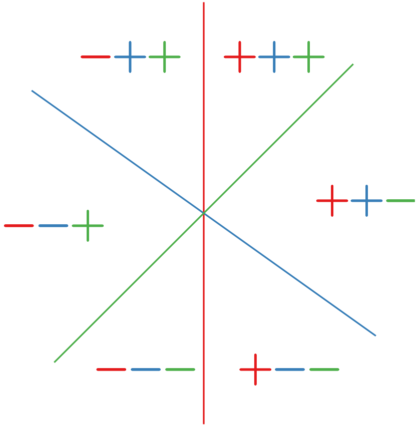
$n \approx 20000$
$d \approx 200 - 500$

OpenImages Dataset
object recognition

$n \approx 9000$
$d \approx 200 - 500$

$n = 3$

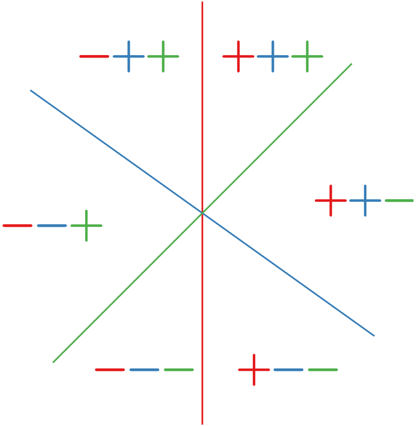
$d = 2$



some label configurations are *unargmaxable*!

$n = 3$

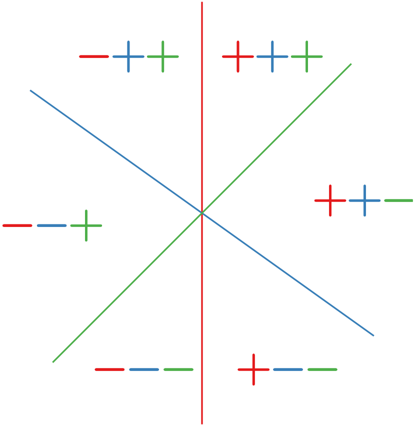
$d = 2$



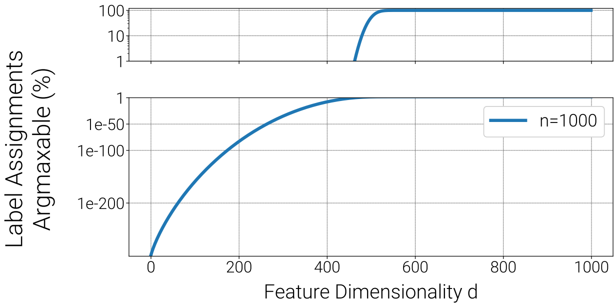
$$\nexists \mathbf{x} : \operatorname*{argmax}_{\mathbf{y}'} p(\mathbf{y}' \mid \mathbf{x}; \mathbf{W}) = \mathbf{y}$$

$n = 3$

$d = 2$



−+− **and** +−+ **are *unargmaxable*!**



exponentially **many configurations are unargmaxable**

but real data is sparse...

K-active labels

clinically annotated text

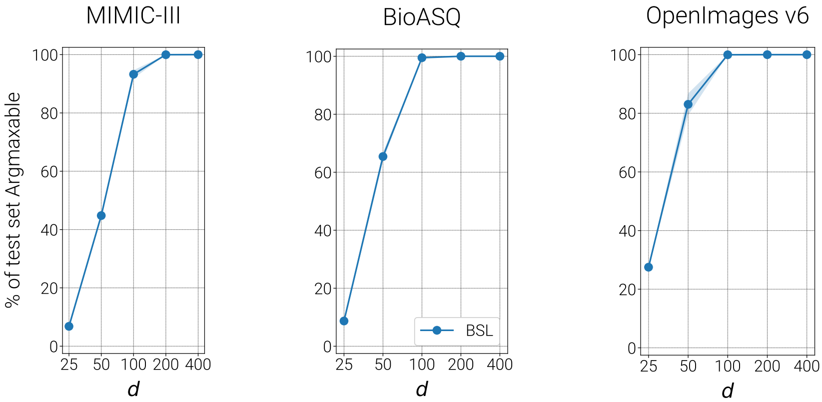
$n \approx 9000$
$K = 80$

large-scale biomedical question answering

$n \approx 20000$
$K = 50$

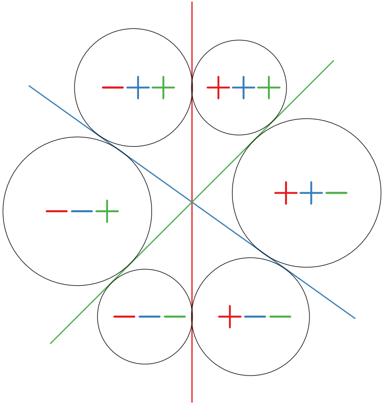
OpenImages Dataset
object recognition

$n \approx 9000$
$K = 50$

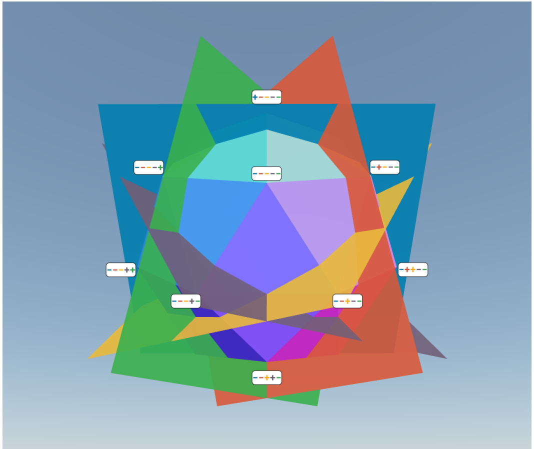


even sparse label configurations are unargmaxable

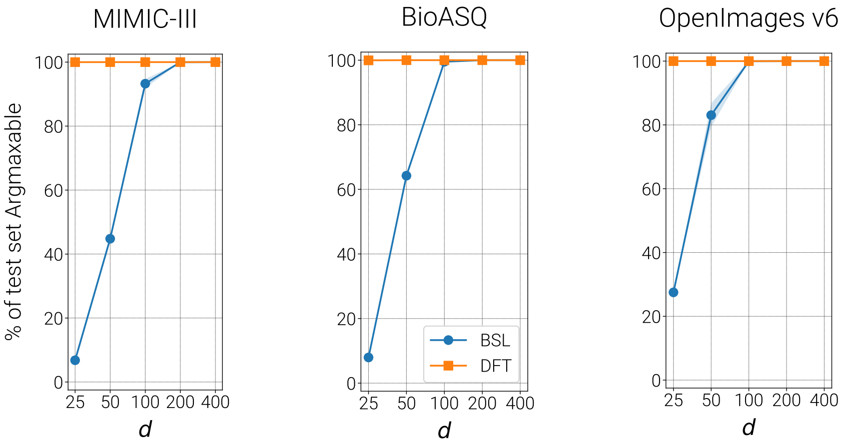
how do we know?



we provide a Chebyshev LP to *verify argmaxability*

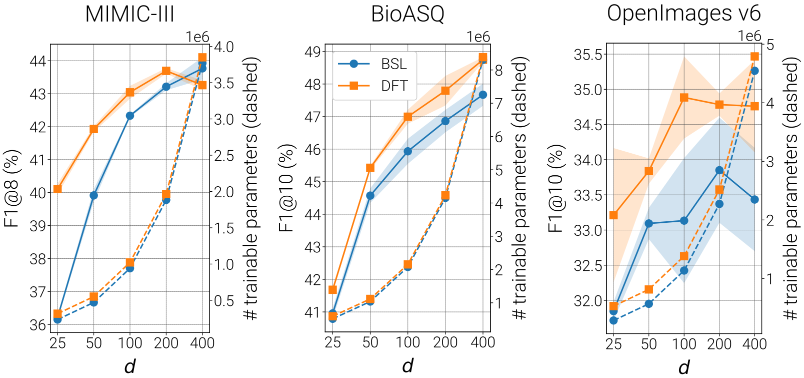


and a new differentiable layer to *guarantee argmaxability*
for K-active label configurations



and a new differentiable layer to guarantee argmaxability

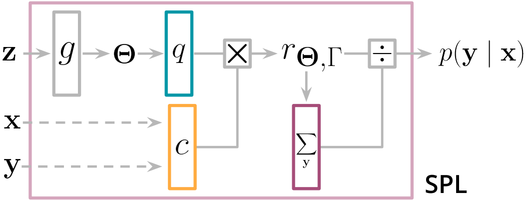
based on the DFT

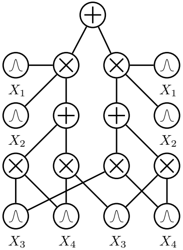


with comparable or better performance

part I how to satisfy wanted constraints in NNs by design

part II how the design of NNs implicitly poses unwanted constraints





Ask me anything!