Probabilistic Circuits
a language for tractable models

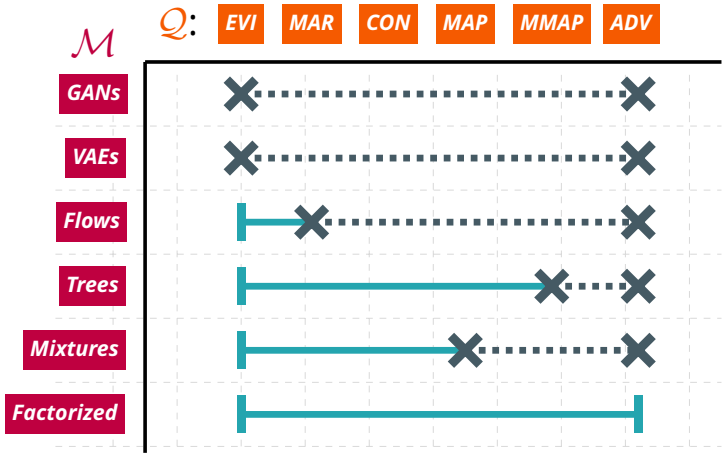
antonio vergari (he/him)

@tetraduzione

6-7th Mar 2024 - **Advanced Probabilistic Modeling** - *University of Trento*

in the previous episodes...

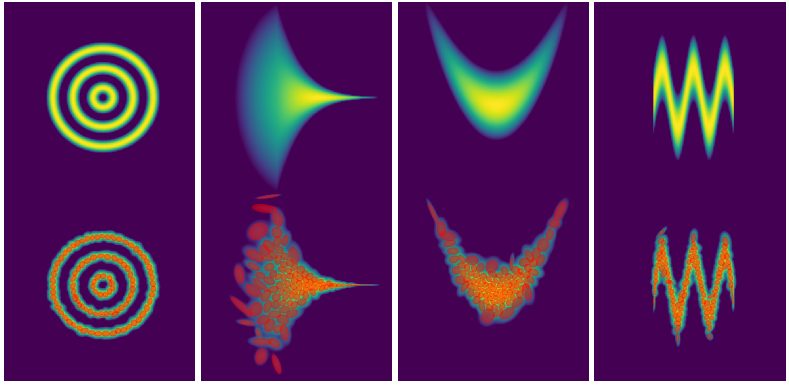
#1 probabilistic reasoning

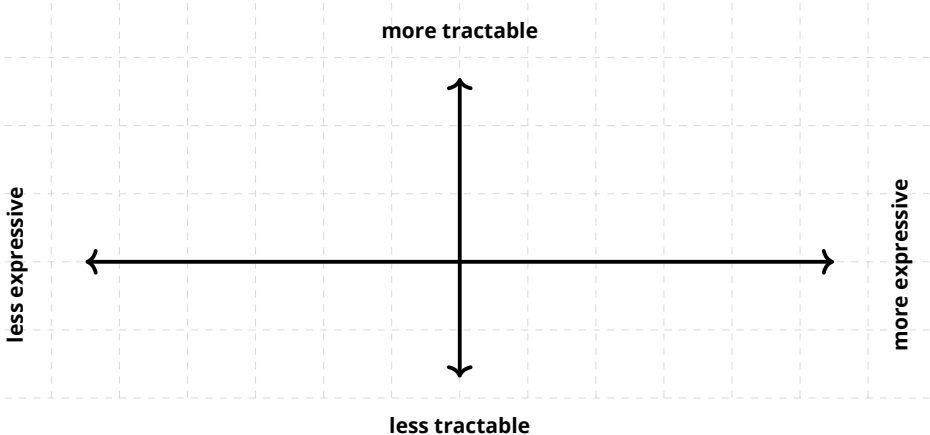


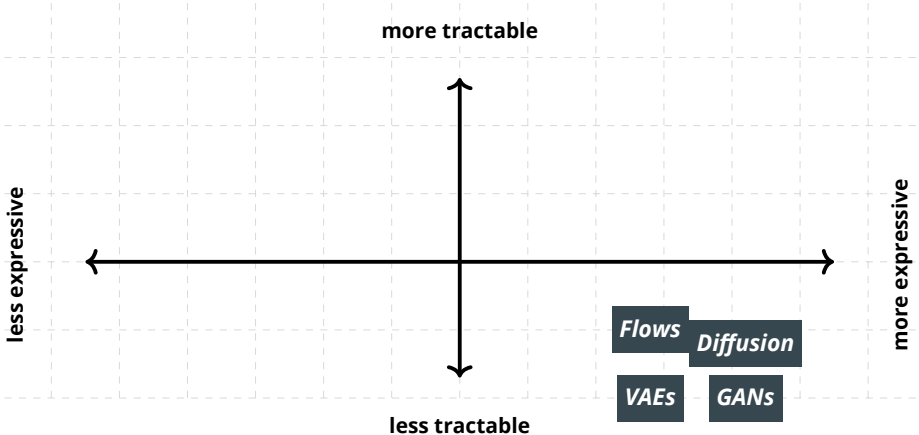
tractable bands

#2 mixture models rock!

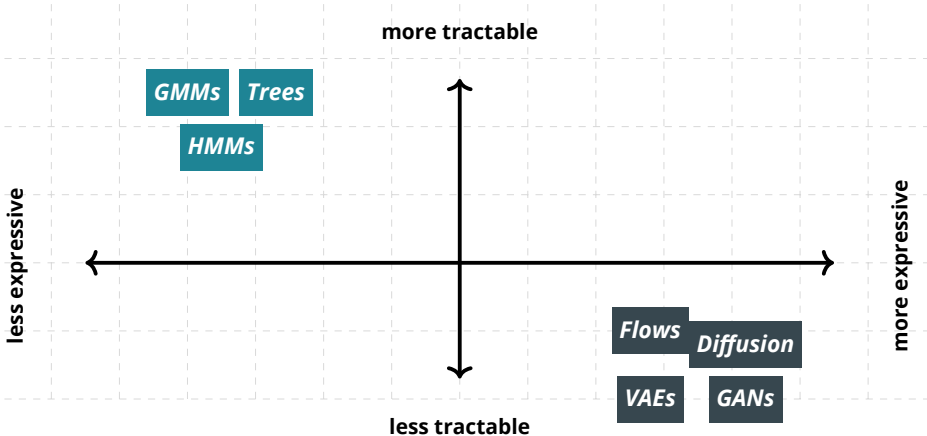

Assumes independence
Assumes independence
No independence!



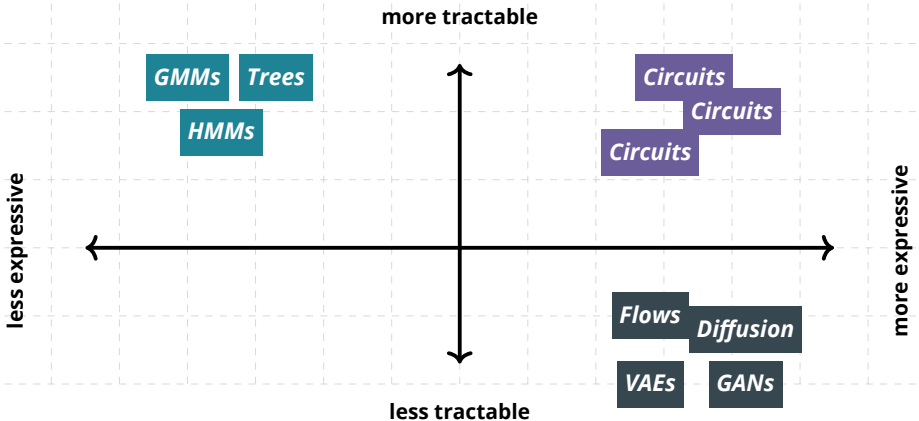

more tractable
less expressive
more expressive
less tractable



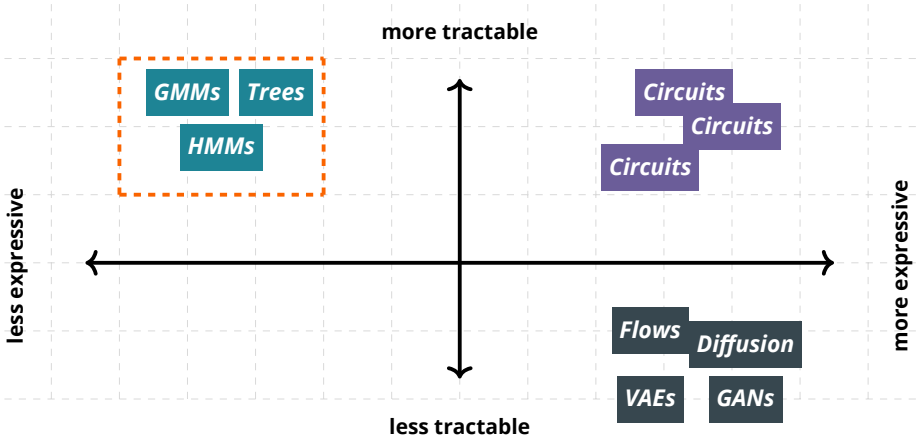
Expressive models are not much tractable...



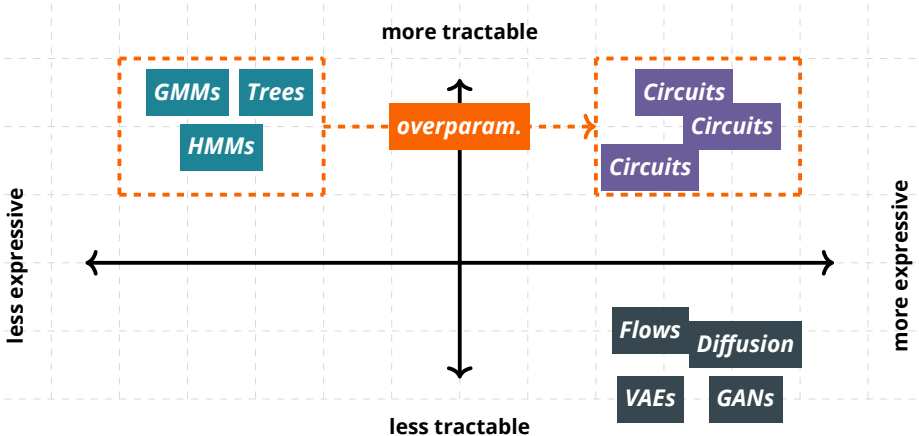
Tractable models are not that expressive...



Circuits can be both expressive and tractable!



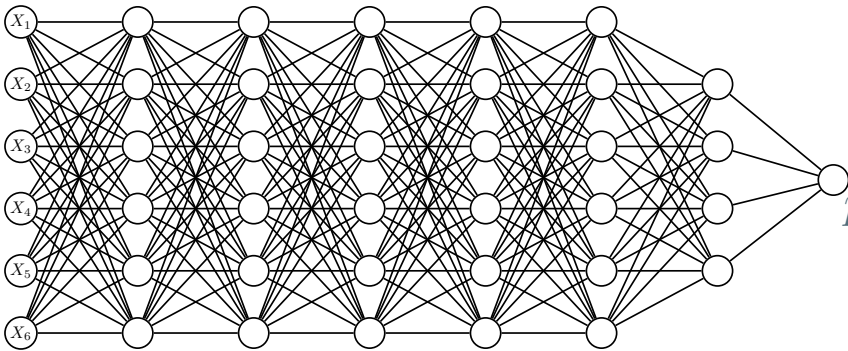
Start simple...



then make it more expressive!


MOGWAI
HA PGMs WILL NEVER DIE,
BUT YOU WILL.

$\mathbf{x}$



$p(\mathbf{x})$
$p(\mathbf{y} \mid \mathbf{x})$

...but what about neural networks?

...so what's the difference?

NNs are ***graphs*** (and ***can encode joint/conditional distributions***), but ...

	PGMs	***Neural Networks***
Nodes:	random variables	unit of computations
Edges:	dependencies	order of execution
Inference:	conditioning	feedforward pass
	elimination	backward pass
	message passing	

...so what's the difference?

NNs are ***graphs*** (and ***can encode joint/conditional distributions***), but ...

	PGMs	***Neural Networks***
Nodes:	random variables	unit of computations
Edges:	dependencies	order of execution
Inference:	conditioning	feedforward pass
	elimination	backward pass
	message passing	

$\Rightarrow$ *they are **computational graphs***

Goal

"Can we find a middle ground between these two representations?"

Goal

"Can we design computational graphs that efficiently encode inference procedures in PGMs?"

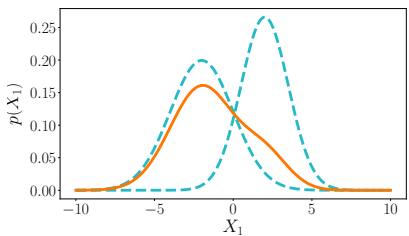
Goal

"Can we design computational graphs that efficiently encode inference procedures in PGMs?"

$\Rightarrow$ ***yes! with circuits!***

GMMs

as computational graphs

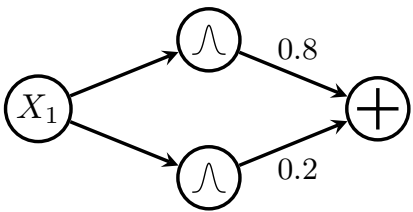


$$p(X) = w_1 \cdot p_1(X_1) + w_2 \cdot p_2(X_1)$$

$\Rightarrow$ *translating inference to data structures...*

GMMs

as computational graphs

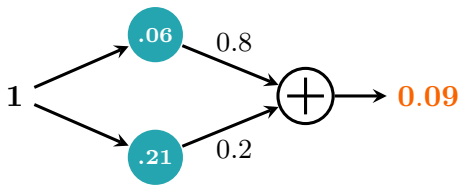


$$p(X_1) = 0.2 \cdot p_1(X_1) + 0.8 \cdot p_2(X_1)$$

$\Rightarrow$ *...e.g., as a weighted sum unit over Gaussian input distributions*

GMMs

as computational graphs

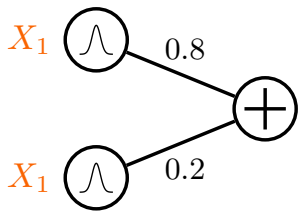


$$p(X = 1) = 0.2 \cdot p_1(X_1 = 1) + 0.8 \cdot p_2(X_1 = 1)$$

$\Rightarrow$ *inference = feedforward evaluation*

GMMs

as computational graphs



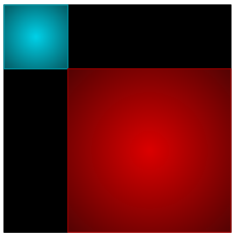
A simplified notation:

⇒ ***scopes*** *attached to inputs*

⇒ *edge directions omitted*

GMMs

as computational graphs

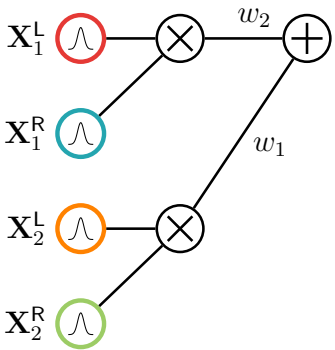


$$p(\mathbf{X}) = w_1 \cdot p_1(\mathbf{X}_1^{\mathsf{L}}) \cdot p_1(\mathbf{X}_1^{\mathsf{R}}) + w_2 \cdot p_2(\mathbf{X}_2^{\mathsf{L}}) \cdot p_2(\mathbf{X}_2^{\mathsf{R}})$$

$\Rightarrow$ *local factorizations...*

GMMs

as computational graphs



$$p(\mathbf{X}) = w_1 \cdot p_1(\mathbf{X}_1^{\mathsf{L}}) \cdot p_1(\mathbf{X}_1^{\mathsf{R}}) + w_2 \cdot p_2(\mathbf{X}_2^{\mathsf{L}}) \cdot p_2(\mathbf{X}_2^{\mathsf{R}})$$

$\Rightarrow$ *...are product units*

Probabilistic Circuits (PCs)

A grammar for tractable computational graphs

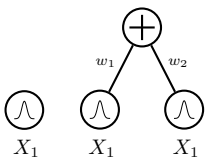
I. *A simple tractable function is a circuit*

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*



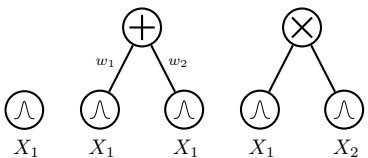
Probabilistic Circuits (PCs)

A grammar for tractable computational graphs

I. *A simple tractable function is a circuit*

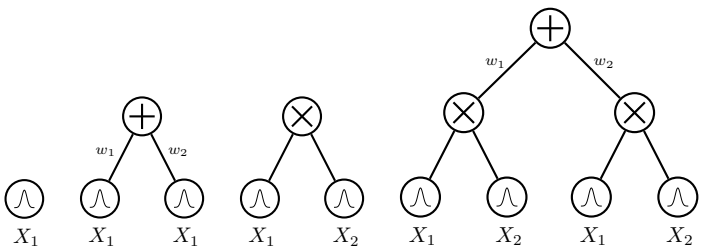
II. *A weighted combination of circuits is a circuit*

III. *A product of circuits is a circuit*



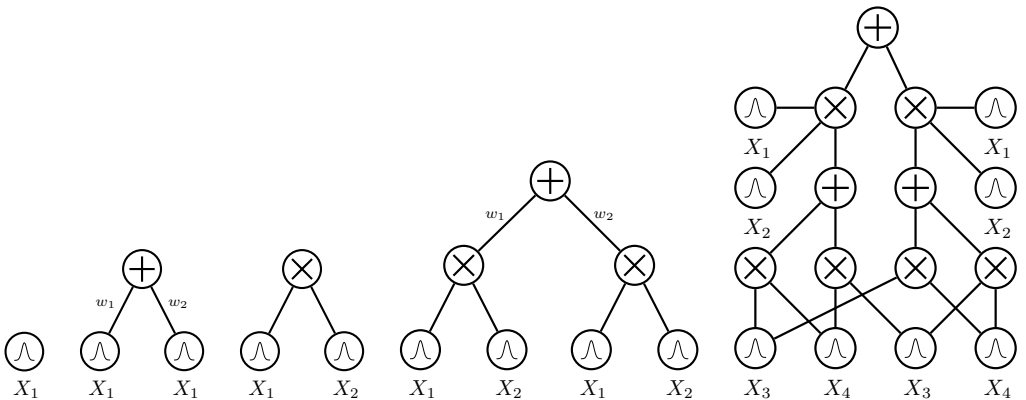
Probabilistic Circuits (PCs)

A grammar for tractable computational graphs



Probabilistic Circuits (PCs)

A grammar for tractable computational graphs



Building PCs in Python with SPFlow

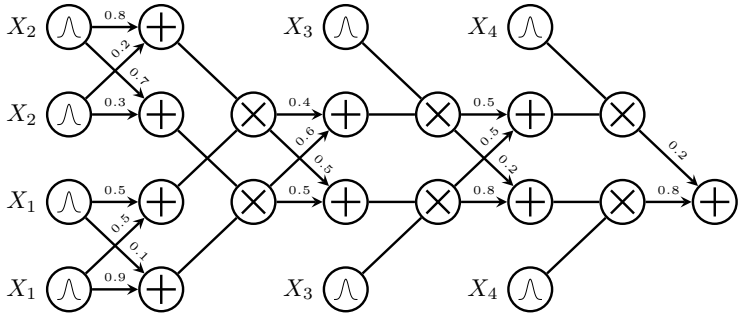

```
import spn.structure.leaves.parametric.Parametric as param
from param import Categorical, Gaussian

PC = 0.4 * (Categorical(p=[0.2, 0.8], scope=0) *
             (0.3 * (Gaussian(mean=1.0, stdev=1.0, scope=1) *
                     Categorical(p=[0.4, 0.6], scope=2))
            + 0.7 * (Gaussian(mean=-1.0, stdev=1.0, scope=1) *
                     Categorical(p=[0.6, 0.4], scope=2)))) \
    + 0.6 * (Categorical(p=[0.2, 0.8], scope=0) *
             Gaussian(mean=0.0, stdev=0.1, scope=1) *
             Categorical(p=[0.4, 0.6], scope=2))
```

Molina et al., "SPFlow: An easy and extensible library for deep probabilistic learning using sum-product networks", arXiv preprint arXiv:1901.03704, 2019

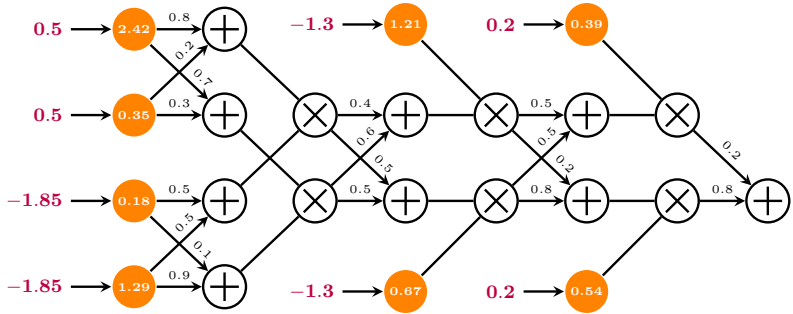
Probabilistic queries = **feedforward** evaluation

$$p(X_1 = -1.85, X_2 = 0.5, X_3 = -1.3, X_4 = 0.2)$$



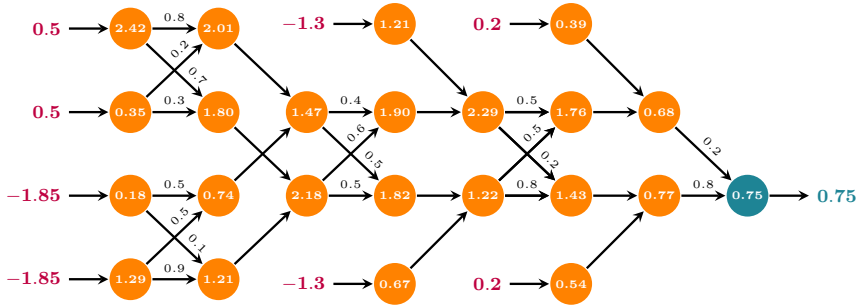
Probabilistic queries = **feedforward** evaluation

$$p(X_1 = -1.85, X_2 = 0.5, X_3 = -1.3, X_4 = 0.2)$$



Probabilistic queries = feedforward evaluation

$$p(X_1 = -1.85, X_2 = 0.5, X_3 = -1.3, X_4 = 0.2) = 0.75$$



...why PCs?

1. A grammar for tractable models

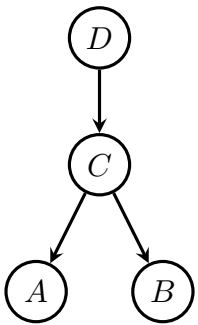
One formalism to represent many probabilistic and logical models

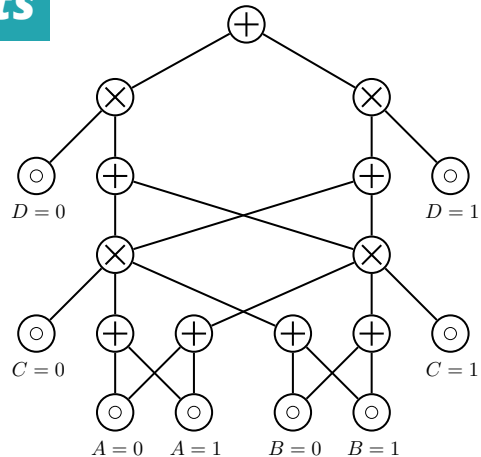
$\Rightarrow$ *#HMMs #Trees #XGBoost, Tensor Networks, ...*

and other *PGMs*...

From PGMs to circuits

via compilation

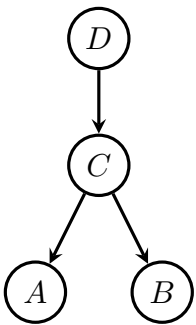




From PGMs to circuits

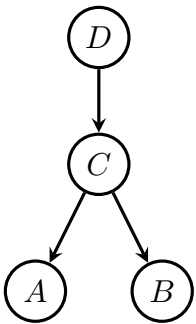
via compilation

Bottom-up **compilation**: starting from leaves...



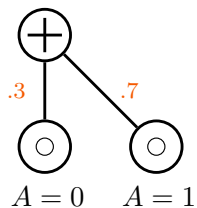
From PGMs to circuits

via compilation



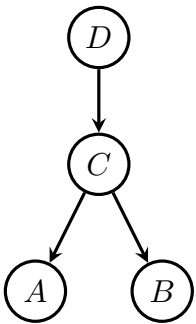
...compile a leaf CPT

$p(A|C=0)$



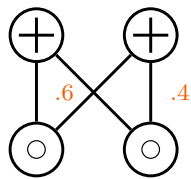
From PGMs to circuits

via compilation



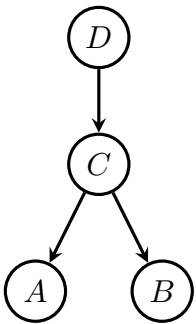
...compile a leaf CPT

$p(A|C=1)$

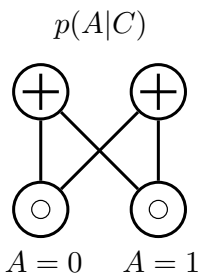


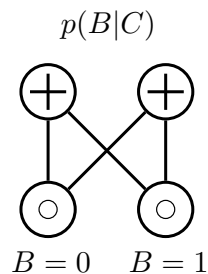
From PGMs to circuits

via compilation



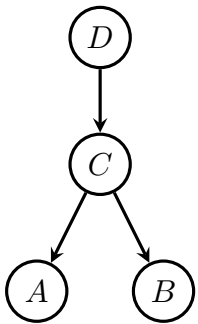
...compile a leaf CPT...for all leaves...



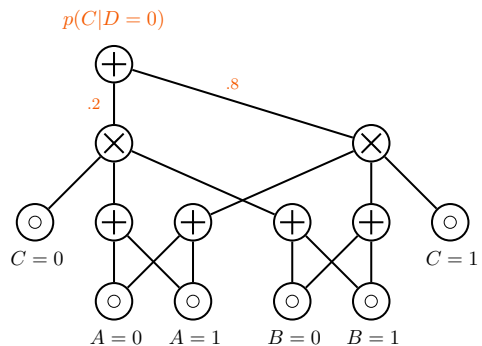


From PGMs to circuits

via compilation

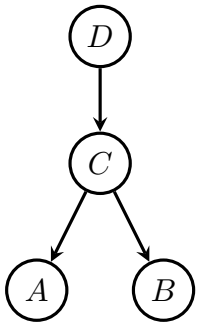


...and recurse over parents...

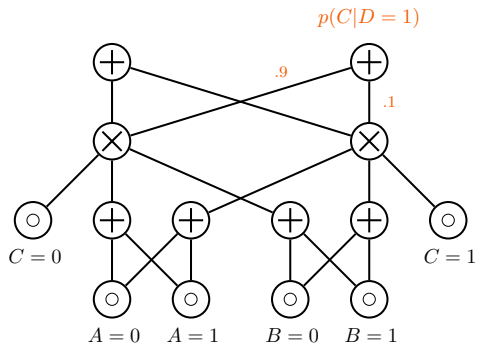


From PGMs to circuits

via compilation

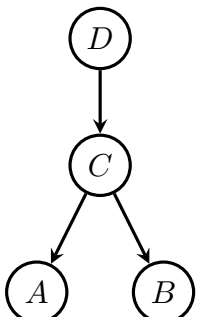


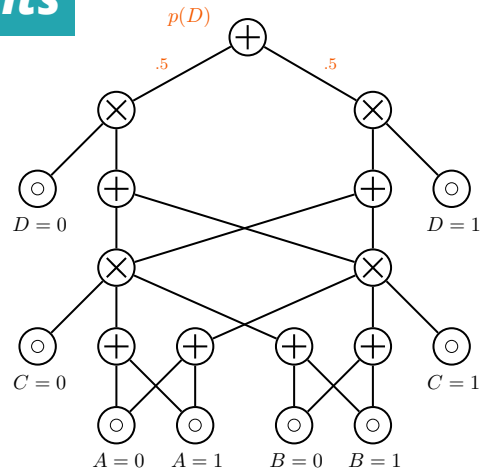
...while reusing previously compiled nodes!...



From PGMs to circuits

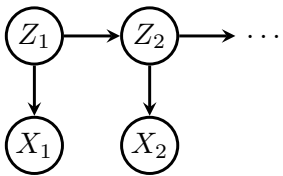
via compilation

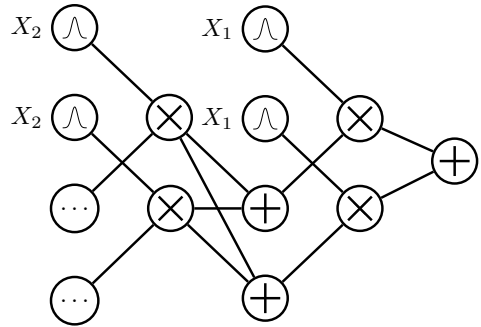




HMMs

as computational graphs





...why PCs?

1. A grammar for tractable models

One formalism to represent many probabilistic and logical models

$\Rightarrow$ *#HMMs #Trees #XGBoost, Tensor Networks, ...*

and other *PGMs*...

...why PCs?

1. A grammar for tractable models

One formalism to represent many probabilistic and logical models

$\Rightarrow$ *#HMMs #Trees #XGBoost, Tensor Networks, ...*

and other *PGMs*...

2. Expressiveness

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

How expressive?

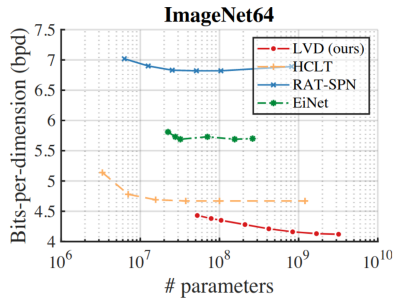
Dataset	Sparse PC (ours)	HCLT	RatSPN	IDF	BitSwap	BB-ANS	McBits
MNIST	**1.14**	1.20	1.67	1.90	1.27	1.39	1.98
EMNIST(MNIST)	**1.52**	1.77	2.56	2.07	1.88	2.04	2.19
EMNIST(Letters)	**1.58**	1.80	2.73	1.95	1.84	2.26	3.12
EMNIST(Balanced)	**1.60**	1.82	2.78	2.15	1.96	2.23	2.88
EMNIST(ByClass)	**1.54**	1.85	2.72	1.98	1.87	2.23	3.14
FashionMNIST	**3.27**	3.34	4.29	3.47	3.28	3.66	3.72

competitive with Flows and VAEs!

Dang, Liu, and Broeck, "Sparse Probabilistic Circuits via Pruning and Growing", NeurIPS, 2022

How scalable?

Dataset	TPMs				DGMs		
	LVD (ours)	HCLT	EiNet	RAT-SPN	Glow	RealNVP	BIVA
ImageNet32	**4.39**±0.01	4.82	5.63	6.90	4.09	4.28	3.96
ImageNet64	**4.12**±0.00	4.67	5.69	6.82	3.81	3.98	-
CIFAR	**4.38**±0.02	4.61	5.81	6.95	3.35	3.49	3.08



up to billions of parameters

Liu, Zhang, and Broeck, "Scaling Up Probabilistic Circuits by Latent Variable Distillation", arXiv preprint, 2022

...why PCs?

1. A grammar for tractable models

One formalism to represent many probabilistic and logical models

$\Rightarrow$ *#HMMs #Trees #XGBoost, Tensor Networks, ...*

and other *PGMs*...

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***

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

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

$\Rightarrow$ *tractable partition function*

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

Structural properties

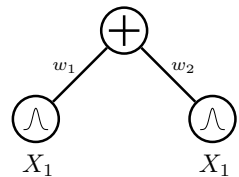
smoothness

decomposability

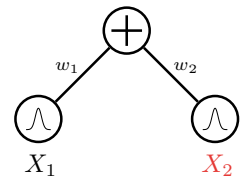
compatibility

determinism

the inputs of sum units are defined over the same variables



***smooth* circuit**



***non-smooth* circuit**

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

Structural properties

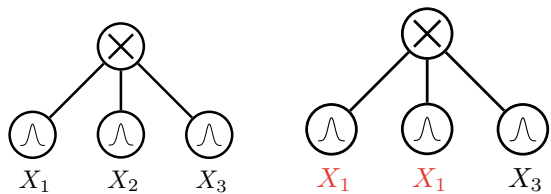
smoothness

decomposability

compatibility

determinism

the inputs of prod units are defined over disjoint variable sets

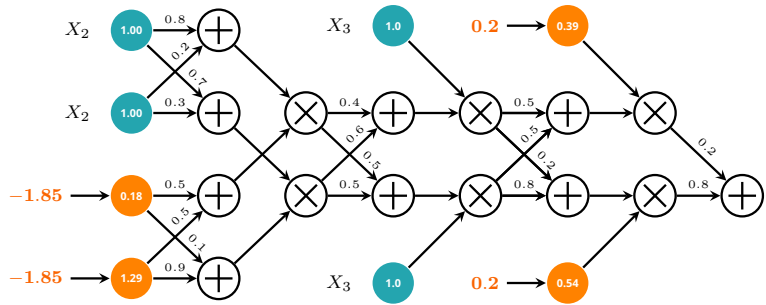


decomposable circuit ***non-decomposable circuit***

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

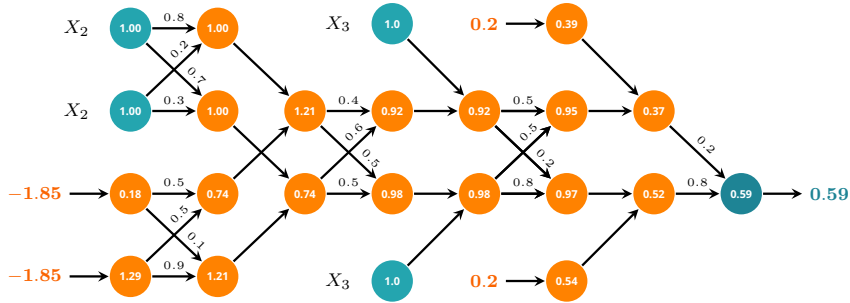
Probabilistic queries = **feedforward** evaluation

$$p(X_1 = -1.85, X_4 = 0.2)$$



Probabilistic queries = **feedforward** evaluation

$$p(X_1 = -1.85, X_4 = 0.2)$$



smooth + *decomposable* circuits = ...

Computing arbitrary integrations (or summations)

$\Rightarrow$ *linear in circuit size!*

E.g., suppose we want to compute Z:

$$\int \boldsymbol{p}(\mathbf{x}) d\mathbf{x}$$

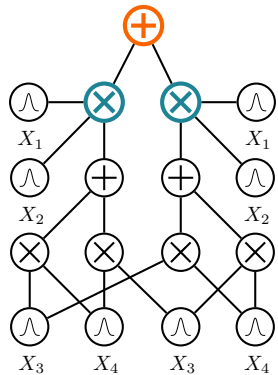
smooth + ***decomposable*** circuits = ...

If $\boldsymbol{p}(\mathbf{x}) = \sum_i w_i \boldsymbol{p}_i(\mathbf{x})$, (***smoothness***):

$$\int \boldsymbol{p}(\mathbf{x}) d\mathbf{x} = \int \sum_i w_i \boldsymbol{p}_i(\mathbf{x}) d\mathbf{x} =$$

$$= \sum_i w_i \int \boldsymbol{p}_i(\mathbf{x}) d\mathbf{x}$$

$\Longrightarrow$ *integrals are "pushed down" to inputs*

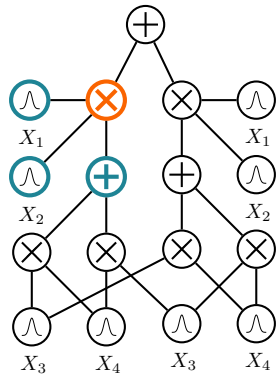


smooth + ***decomposable*** circuits = ...

If $p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{x})p(\mathbf{y})p(\mathbf{z})$, (***decomposability***):

$$\int\int\int p(\mathbf{x}, \mathbf{y}, \mathbf{z}) d\mathbf{x} d\mathbf{y} d\mathbf{z} =$$
$$= \int\int\int p(\mathbf{x})p(\mathbf{y})p(\mathbf{z}) d\mathbf{x} d\mathbf{y} d\mathbf{z} =$$
$$= \int p(\mathbf{x}) d\mathbf{x} \int p(\mathbf{y}) d\mathbf{y} \int p(\mathbf{z}) d\mathbf{z}$$

$\Longrightarrow$ *integrals decompose into easier ones*

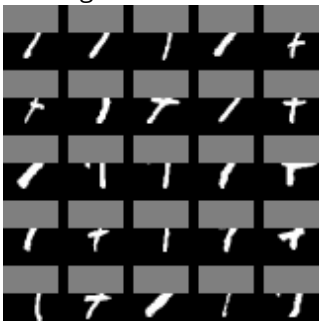


Tractable inference on PCs

Einsum networks

Original

Missing



Conditional sample

Peharz et al., "Einsum Networks: Fast and Scalable Learning of Tractable Probabilistic Circuits", , 2020

Reasoning about ML models

q_1 *"What is the probability of a treatment for a patient with* ***unavailable records****?"*

q_2 *"How* ***fair*** *is the prediction is a certain protected attribute changes?"*

q_3 *"Can we certify no* ***adversarial examples*** *exist?"*

Reasoning about ML models

q1 $\int p(\mathbf{x}_o, \mathbf{x}_m) d\mathbf{X}_m$
(missing values)

q2 $\mathbb{E}_{\mathbf{x}_c \sim p(\mathbf{x}_c | X_s=0)} [f_0(\mathbf{x}_c)] - \mathbb{E}_{\mathbf{x}_c \sim p(\mathbf{x}_c | X_s=1)} [f_1(\mathbf{x}_c)]$
(fairness)

q3 $\mathbb{E}_{\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_D)} [f(\mathbf{x} + \mathbf{e})]$
(adversarial robust.)

...in the language of probabilities

Reasoning about ML models

q1 $\int p(\mathbf{x}_o, \mathbf{x}_m) d\mathbf{X}_m$
(missing values)

q2 $\mathbb{E}_{\mathbf{x}_c \sim p(\mathbf{x}_c | X_s = 0)} [f_0(\mathbf{x}_c)] - \mathbb{E}_{\mathbf{x}_c \sim p(\mathbf{x}_c | X_s = 1)} [f_1(\mathbf{x}_c)]$
(fairness)

q3 $\mathbb{E}_{\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_D)} [f(\mathbf{x} + \mathbf{e})]$
(adversarial robust.)

it is crucial we compute them exactly and in polytime!

Reasoning about ML models

q1 $\int p(\mathbf{x}_o, \mathbf{x}_m) d\mathbf{X}_m$
(missing values)

q2 $\mathbb{E}_{\mathbf{x}_c \sim p(\mathbf{x}_c | X_s=0)} [f_0(\mathbf{x}_c)] - \mathbb{E}_{\mathbf{x}_c \sim p(\mathbf{x}_c | X_s=1)} [f_1(\mathbf{x}_c)]$
(fairness)

q3 $\mathbb{E}_{\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_D)} [f(\mathbf{x} + \mathbf{e})]$
(adversarial robust.)

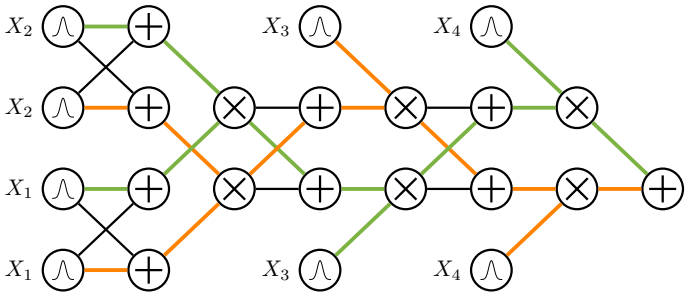
it is crucial we compute them tractably!

Which structural properties

for complex reasoning

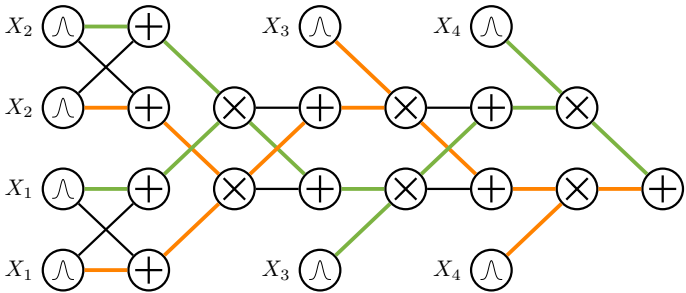
smooth + decomposable

Expressive efficiency



$$p(\mathbf{x}) = \sum_{\mathcal{T}} \left(\prod_{w_j \in \mathbf{w}_{\mathcal{T}}} w_j \right) \prod_{l \in \text{leaves}(\mathcal{T})} p_l(\mathbf{x})$$

Expressive efficiency



an exponential number of mixture components!

...wait!

“Are all PGMs circuits?”

and/or

“Are all circuits PGMs?”

...not so easy!

1. Marginal inference in PGMs is exponential in the treewidth!

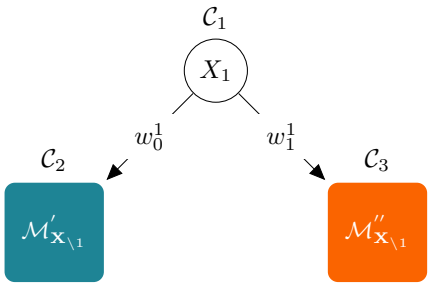
but PCs can exploit context specific independence

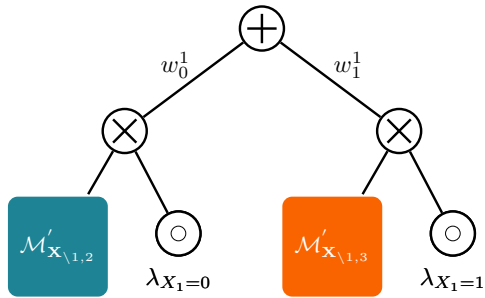
$$\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid Z = z_1$$

but

$$\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid Z = z_2$$

Decision trees as PCs...





...not so easy!

1. Marginal inference in PGMs is exponential in the treewidth!
but PCs can exploit **context specific independence**

2. We do not know how to compile exactly an arbitrary PGM over continuous vars!
we need to extend the language of PCs to **integral units**

Probabilistic Integral Circuits

Gennaro Gala[1] Cassio de Campos[1] Robert Peharz[1,2] Antonio Vergari[3] Erik Quaeghebeur[1]

Which structural properties

for complex reasoning

smooth + decomposable

??????

Adversarial smoothing

Certify robustness for inputs $\mathbf{x}$ by ***smoothing*** it by computing

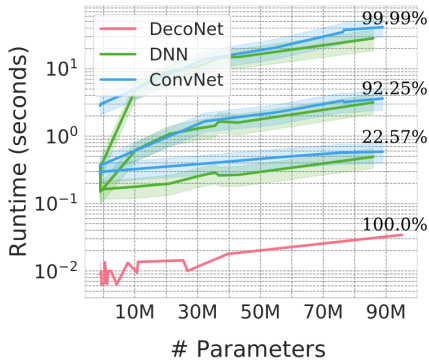
$$g_\sigma(\mathbf{x}) = \mathbb{E}_{\mathbf{e}\sim\mathcal{N}(0,\sigma\mathbf{I})}\left[f(\mathbf{x}+\mathbf{e})\right]$$

Subramani et al., "Exact and Efficient Adversarial Robustness with Decomposable Neural Networks", Uncertainty in Deep Learning Workshop ICML, 2021

Adversarial smoothing

Certify robustness for inputs $\mathbf{x}$ by ***smoothing*** it by computing

$$g_\sigma(\mathbf{x}) = \mathbb{E}_{\mathbf{e}\sim\mathcal{N}(0,\sigma\mathbf{I})}\left[f(\mathbf{x}+\mathbf{e})\right]$$



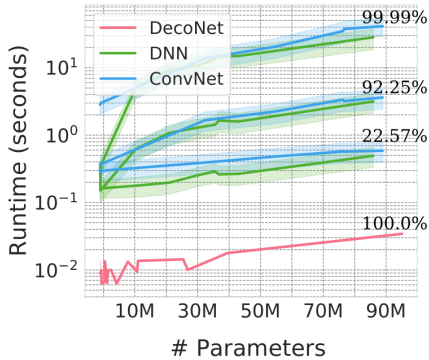
Subramani et al., "Exact and Efficient Adversarial Robustness with Decomposable Neural Networks", Uncertainty in Deep Learning Workshop ICML, 2021

Adversarial smoothing

Certify robustness for inputs $\mathbf{x}$ by ***smoothing*** it by computing

$$g_\sigma(\mathbf{x}) = \mathbb{E}_{\mathbf{e}\sim\mathcal{N}(0,\sigma\mathbf{I})}\left[f(\mathbf{x}+\mathbf{e})\right]$$

in a single feed-forward evaluation, if we ***impose some structure*** over a computational graph



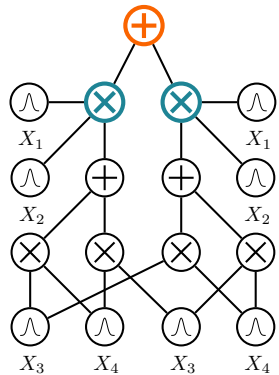
Subramani et al., "Exact and Efficient Adversarial Robustness with Decomposable Neural Networks", Uncertainty in Deep Learning Workshop ICML, 2021

decomposable circuits = tractable *adv smoothing*

If $f(\mathbf{x}) = \sum_i w_i f_i(\mathbf{x})$:

$$\int \mathcal{N}(\mathbf{e}) f(\mathbf{x}+\mathbf{e}) d\mathbf{e} = \sum_i w_i \mathbb{E}_{\mathcal{N}(\mathbf{e})}[f_i(\mathbf{x}+\mathbf{e})]$$

$\Longrightarrow$ *expectations are "pushed down" to inputs*



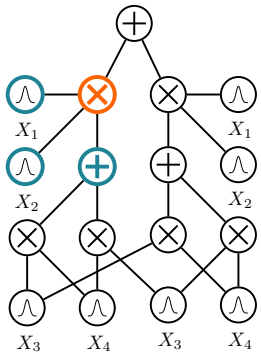
decomposable circuits = tractable *adv smoothing*

If $f(\mathbf{x}, \mathbf{y}, \mathbf{z}) = f(\mathbf{x}) f(\mathbf{y}) f(\mathbf{z})$, (***decomposability***):

$$\int \mathcal{N}(\mathbf{e_x}) \mathcal{N}(\mathbf{e_y}) \mathcal{N}(\mathbf{e_y}) f(\mathbf{x} + \mathbf{e_x}, \mathbf{y} + \mathbf{e_y}, \mathbf{z} + \mathbf{e_z}) d\mathbf{e_x} d\mathbf{e_y} d\mathbf{e_z}$$

$$\mathbb{E}_{\mathbf{e_x}}[f(\mathbf{x} + \mathbf{e_x})] \cdot \mathbb{E}_{\mathbf{e_y}}[f(\mathbf{y} + \mathbf{e_y})] \cdot \mathbb{E}_{\mathbf{e_z}}[f(\mathbf{z} + \mathbf{e_z})]$$

$\Longrightarrow$ *expectations decompose into easier ones*



Which structural properties

for complex reasoning

smooth + decomposable

decomposable

Which structural properties

for complex reasoning

smooth + decomposable

???????

decomposable

General expectations

Integrals involving two or more functions:

$$\int p(\mathbf{x}) f(\mathbf{x}) d\,\mathbf{X}$$

General expectations

Integrals involving two or more functions:

$$\int p(\mathbf{x}) f(\mathbf{x}) d\,\mathbf{X}$$

represent both p and f as circuits...but with which structural properties? E.g.,

General expectations

Integrals involving two or more functions:

$$\int p(\mathbf{x}) f(\mathbf{x}) d\,\mathbf{X}$$

represent both p and f as circuits...but with which structural properties? E.g.,

$$\mathbb{E}_{\mathbf{x}_c \sim p(\mathbf{X}_c | X_s = 0)} \left[f_0(\mathbf{x}_c) \right] - \mathbb{E}_{\mathbf{x}_c \sim p(\mathbf{X}_c | X_s = 1)} \left[f_1(\mathbf{x}_c) \right]$$

Structural properties

smoothness

decomposability

compatibility

determinism

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

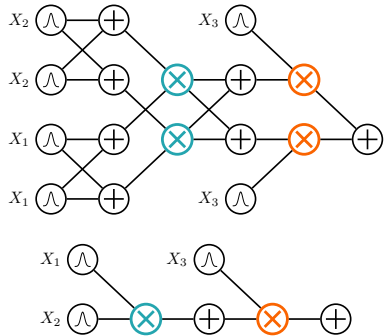
Structural properties

smoothness

decomposability

compatibility

determinism



compatible circuits

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

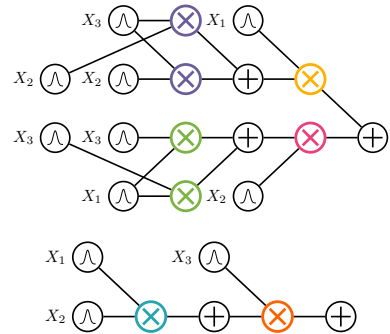
Structural properties

smoothness

decomposability

compatibility

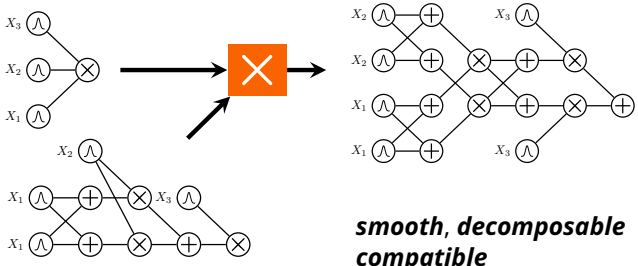
determinism



non-compatible circuits

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

Tractable products



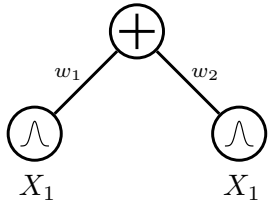
smooth, decomposable compatible

exactly compute $\int \boldsymbol{p}(\mathrm{x})\boldsymbol{f}(\mathrm{x})d\mathrm{X}$ **in time** $O(|\boldsymbol{p}||\boldsymbol{f}|)$

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

Tractable products

$$p(\mathbf{X}) = \sum_{i=1}^{K} w_i p_i(\mathbf{X}), \qquad \sum_{i=1}^{K} w_i = 1, \quad w_i \geq 0,$$



Tractable products

$$p(\mathbf{X}) = \sum_{i=1}^{K} w_i p_i(\mathbf{X}), \qquad \sum_{i=1}^{K} w_i = 1, \quad w_i \geq 0,$$

$$q(\mathbf{X}) = \sum_{j=1}^{L} w'_j q_j(\mathbf{X}), \qquad \sum_{j=1}^{L} w'_j = 1, \quad w'_j \geq 0,$$

Tractable products

$$p(\mathbf{X}) = \sum_{i=1}^{K} w_i p_i(\mathbf{X}), \qquad \sum_{i=1}^{K} w_i = 1, \quad w_i \geq 0,$$

$$q(\mathbf{X}) = \sum_{j=1}^{L} w'_j q_j(\mathbf{X}), \qquad \sum_{j=1}^{L} w'_j = 1, \quad w'_j \geq 0,$$

$$p(\mathbf{X}) \times q(\mathbf{X}) = \sum_{i=1}^{K} \sum_{j=1}^{L} w_i w_j p_i(\mathbf{X}) q_j(\mathbf{X}),$$

cartesian product of “local” mixtures

Which structural properties

for complex reasoning

smooth + decomposable

compatibility

decomposable

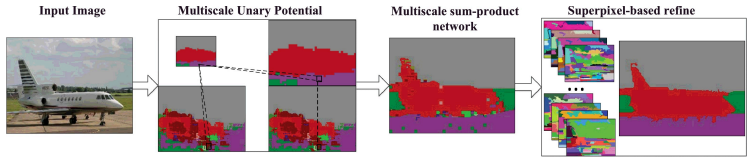
Maximum-a-posteriori (MAP) Inference

aka Most probable explanation (MPE)

E.g., multi-label classification: what are the most likely labels $\mathbf{y}$ for an input $\mathbf{x}$?

$$\underset{\mathbf{y}}{\operatorname{argmax}}\, p(\mathbf{y} \mid \mathbf{x})$$

E.g., image segmentation: what is the most likely latent space for the given pixels?



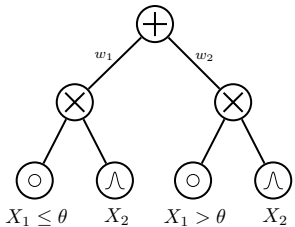
Yuan et al., "Modeling spatial layout for scene image understanding via a novel multiscale sum-product network", Expert Systems with Applications, 2016

Friesen and Domingos, "Submodular Sum-product Networks for Scene Understanding", , 2016

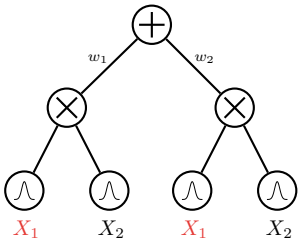
Determinism

aka support-decomposability

A sum unit is deterministic if its inputs have disjoint supports



deterministic circuit



***non*-deterministic circuit**

Darwiche and Marquis, "A knowledge compilation map", Journal of Artificial Intelligence Research, 2002

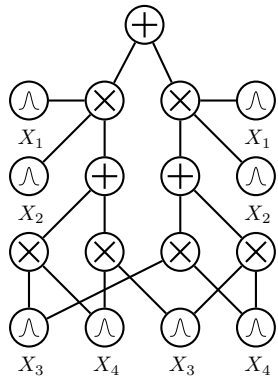
Determinism + **decomposability** = **tractable MAP**

Computing maximization with arbitrary evidence $\mathbf{e}$

$\Rightarrow$ *linear in circuit size!*

E.g., suppose we want to compute:

$$\max_{\mathbf{q}} p(\mathbf{q} \mid \mathbf{e})$$

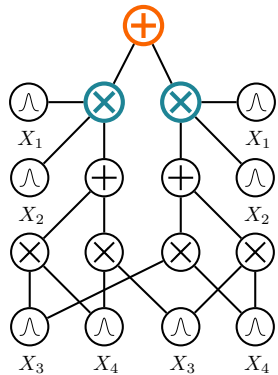


Determinism + **decomposability** = **tractable MAP**

If $\boldsymbol{p}(\mathbf{q}, \mathbf{e}) = \sum_i w_i \boldsymbol{p}_i(\mathbf{q}, \mathbf{e}) = \max_i w_i \boldsymbol{p}_i(\mathbf{q}, \mathbf{e})$,
(***deterministic*** sum unit):

$$\begin{aligned}\max_{\mathbf{q}} \boldsymbol{p}(\mathbf{q}, \mathbf{e}) &= \max_{\mathbf{q}} \sum_i w_i \boldsymbol{p}_i(\mathbf{q}, \mathbf{e}) \\ &= \max_{\mathbf{q}} \max_i w_i \boldsymbol{p}_i(\mathbf{q}, \mathbf{e}) \\ &= \max_i \max_{\mathbf{q}} w_i \boldsymbol{p}_i(\mathbf{q}, \mathbf{e})\end{aligned}$$

$\Rightarrow$ *one non-zero term, thus sum is max*

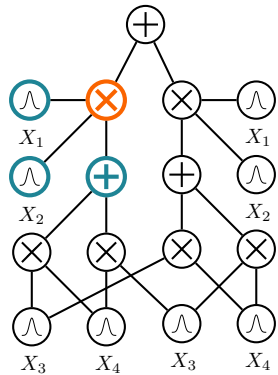


Determinism + **decomposability** = **tractable MAP**

If $p(\mathbf{q}, \mathbf{e}) = p(\mathbf{q_x}, \mathbf{e_x}, \mathbf{q_y}, \mathbf{e_y}) = p(\mathbf{q_x}, \mathbf{e_x})p(\mathbf{q_y}, \mathbf{e_y})$ (***decomposable*** product unit):

$$\begin{aligned}\max_{\mathbf{q}} p(\mathbf{q} \mid \mathbf{e}) &= \max_{\mathbf{q}} p(\mathbf{q}, \mathbf{e}) \\ &= \max_{\mathbf{q_x}, \mathbf{q_y}} p(\mathbf{q_x}, \mathbf{e_x}, \mathbf{q_y}, \mathbf{e_y}) \\ &= \max_{\mathbf{q_x}} p(\mathbf{q_x}, \mathbf{e_x}) \cdot \max_{\mathbf{q_y}} p(\mathbf{q_y}, \mathbf{e_y})\end{aligned}$$

$\Rightarrow$ *solving optimization independently*



Determinism + **decomposability** = **tractable MAP**

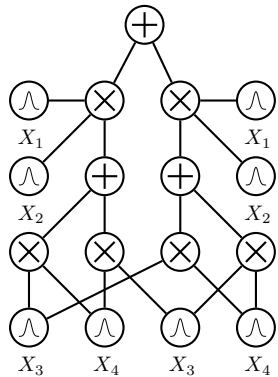
E.g., for $\operatorname{argmax}_{x_1,x_3} p(x_1, x_3 \mid x_2, x_4)$:

turn sum into max units and
input distributions into max distributions

feedforward evaluation for
$\max_{x_1,x_3} p(x_1, x_3 \mid x_2, x_4)$

retrieve max activations in backward pass

compute **MAP states** for X_1 and X_3 at input units



Determinism + **decomposability** = **tractable MAP**

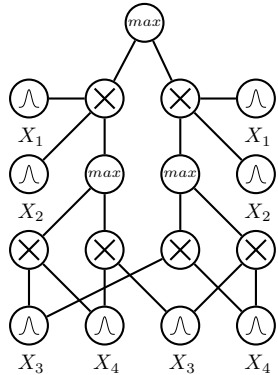
E.g., for $\operatorname{argmax}_{x_1,x_3} p(x_1, x_3 \mid x_2, x_4)$:

turn sum into max units and
input distributions into max distributions

feedforward evaluation for
$\max_{x_1,x_3} p(x_1, x_3 \mid x_2, x_4)$

retrieve max activations in backward pass

compute **MAP states** for X_1 and X_3 at input units



Determinism + **decomposability** = **tractable MAP**

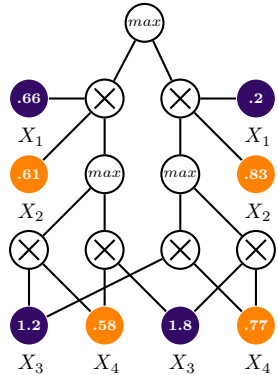
E.g., for $\text{argmax}_{x_1,x_3}\, p(x_1, x_3 \mid x_2, x_4)$:

turn sum into max units and
input distributions into max distributions

feedforward evaluation for
$\max_{x_1,x_3}\, p(x_1, x_3 \mid x_2, x_4)$

retrieve max activations in backward pass

compute **MAP states** for X_1 and X_3 at input units



Determinism + **decomposability** = **tractable MAP**

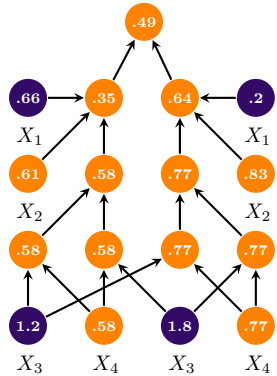
E.g., for $\text{argmax}_{x_1,x_3}\, p(x_1, x_3 \mid x_2, x_4)$:

turn sum into max units and
input distributions into max distributions

feedforward evaluation for
$\max_{x_1,x_3}\, p(x_1, x_3 \mid x_2, x_4)$

retrieve max activations in backward pass

compute **MAP states** for X_1 and X_3 at input units



Determinism + **decomposability** = **tractable MAP**

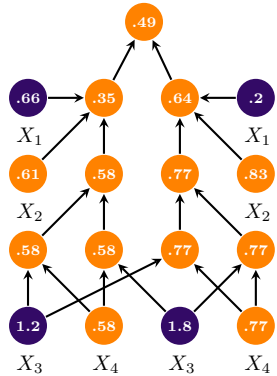
E.g., for $\text{argmax}_{x_1,x_3}\, p(x_1, x_3 \mid x_2, x_4)$:

turn sum into max units and
input distributions into max distributions

feedforward evaluation for
$\max_{x_1,x_3}\, p(x_1, x_3 \mid x_2, x_4)$

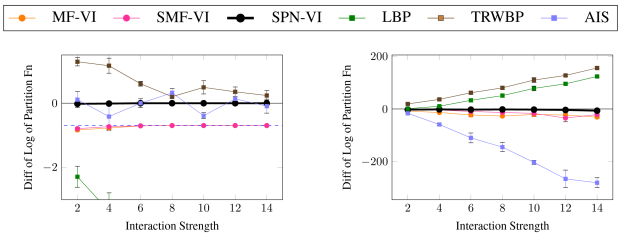
retrieve max activations in backward pass

compute **MAP states** for X_1 and X_3 at input units



Example: Tractable ELBO

Using deterministic and decomposable PCs as expressive variational family $\mathcal{Q}$ for discrete polynomial log-densities, i.e. $\operatorname{argmax}_{q \in \mathcal{Q}} \mathbb{E}_{\mathbf{x} \sim q} [\log w(\mathbf{x})] + \mathbb{H}(q)$



Closed-form computation for the entropy $\mathbb{H}$ *[**Vergari** et al. 2021]*

PCs

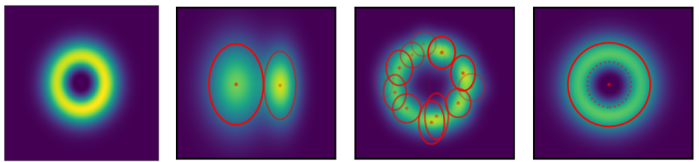
(and more circuits)

everywhere

SUBTRACTIVE MIXTURE MODELS VIA SQUARING: REPRESENTATION AND LEARNING

Lorenzo Loconte[1] **Aleksanteri M. Sladek**[2] **Stefan Mengel**[3]

Martin Trapp[2] **Arno Solin**[2] **Nicolas Gillis**[4] **Antonio Vergari**[1]



GMM ($K = 2$) GMM ($K = 16$) nGMM^2 ($K = 2$)

PGMs with negative parameters!
spotlight at ICLR 2024 (top 5% papers)

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

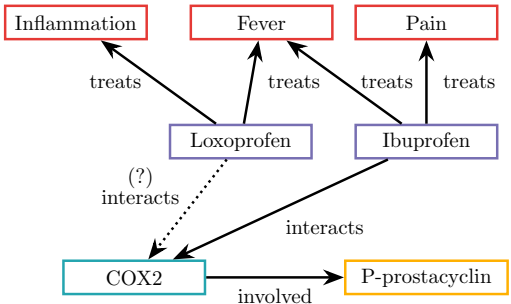
Lorenzo Loconte
University of Edinburgh, UK
l.loconte@sms.ed.ac.uk

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

PCs meet knowledge graph embedding models
oral at NeurIPS 2023 (top 0.6% papers)



Drugs Symptoms

Proteins Functions

⟨Loxoprofen, treats, Inflammation⟩

⟨Ibuprofen, interacts, COX2⟩

⟨COX2, involved, P-prostacyclin⟩

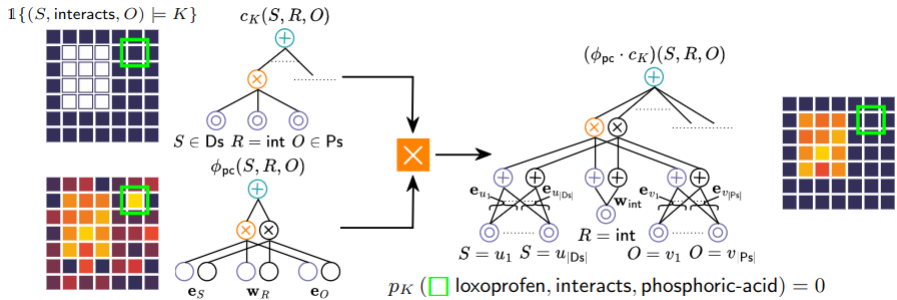
...

Q: ⟨Loxoprofen, interacts, ?⟩

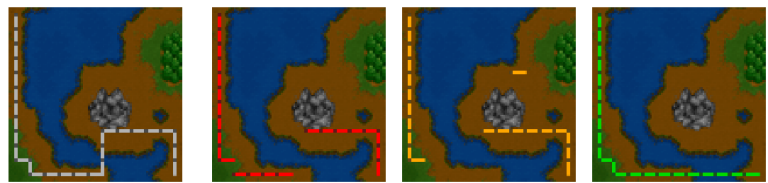
A: ⟨Loxoprofen, interacts, **phosphoric-acid**⟩ !!!

neural link predictors can violate domain constraints

PCs+KGE

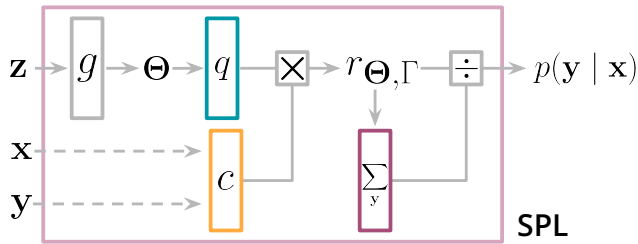

$\mathbb{1}\{(S, \text{interacts}, O) \models K\}$
$c_K(S, R, O)$
$S \in \text{Ds}\ R = \text{int}\ O \in \text{Ps}$
$\phi_{\text{pc}}(S, R, O)$
$\mathbf{e}_S$
$\mathbf{w}_R$
$\mathbf{e}_O$
$(\phi_{\text{pc}} \cdot c_K)(S, R, O)$
$\mathbf{e}_{u_1}$
$\mathbf{e}_{u_{|\text{Ds}|}}$
$\mathbf{w}_{\text{int}}$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

Semantic Probabilistic Layers for Neuro-Symbolic Learning



injecting constraints in deep learning (NeurIPS 2022)

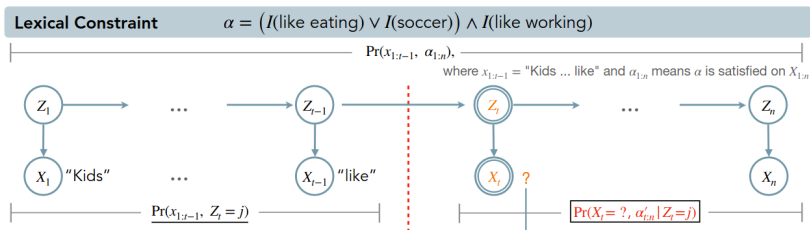
$$p(\mathbf{y} \mid \mathbf{x}) = q_{\Theta}(\mathbf{y} \mid g(\mathbf{z})) \cdot c_{\mathsf{K}}(\mathbf{x}, \mathbf{y}) / \mathcal{Z}(\mathbf{x})$$



efficient and reliable reasoning over constraints

Tractable Control for Autoregressive Language Generation

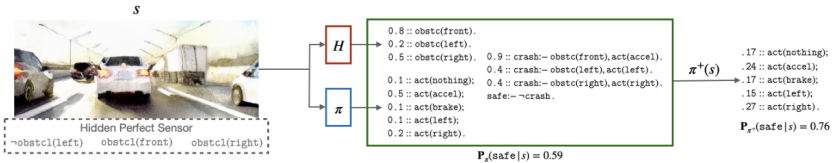
Honghua Zhang [*1] Meihua Dang [*1] Nanyun Peng [1] Guy Van den Broeck [1]



constrained text generation with LLMs (ICML 2023)

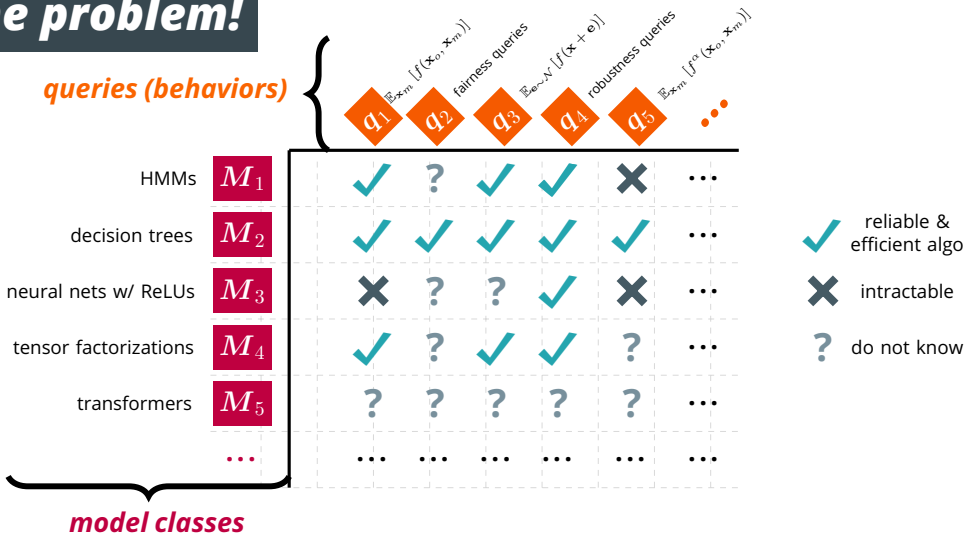
Safe Reinforcement Learning via Probabilistic Logic Shields

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



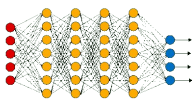
reliable reinforcement learning (AAAI 23)

The problem!


queries (behaviors)
q_1 $\mathbb{E}_{\mathbf{x}_m}[f(\mathbf{x}_o, \mathbf{x}_m)]$
q_2 fairness queries
q_3 $\mathbb{E}_{\mathbf{e}\sim\mathcal{N}}[f(\mathbf{x}+\mathbf{e})]$
q_4 robustness queries
q_5 $\mathbb{E}_{\mathbf{x}_m}[f^a(\mathbf{x}_o, \mathbf{x}_m)]$
HMMs M_1
decision trees M_2
neural nets w/ ReLUs M_3
tensor factorizations M_4
transformers M_5
model classes
reliable & efficient algo
intractable
do not know

"How uncertain will the classifier be if some input is missing?"

behaviors
to be inspected



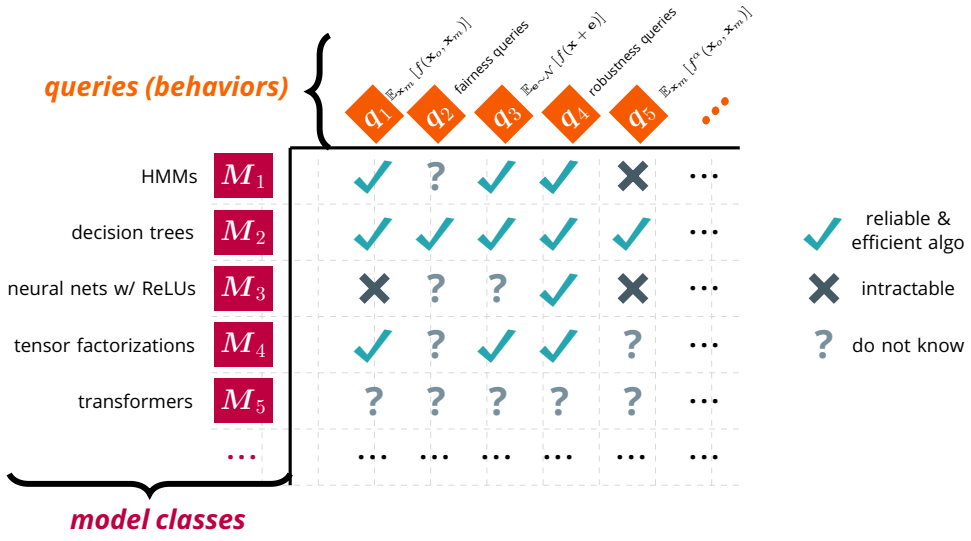
ML models
(classifiers, generative models, ...)

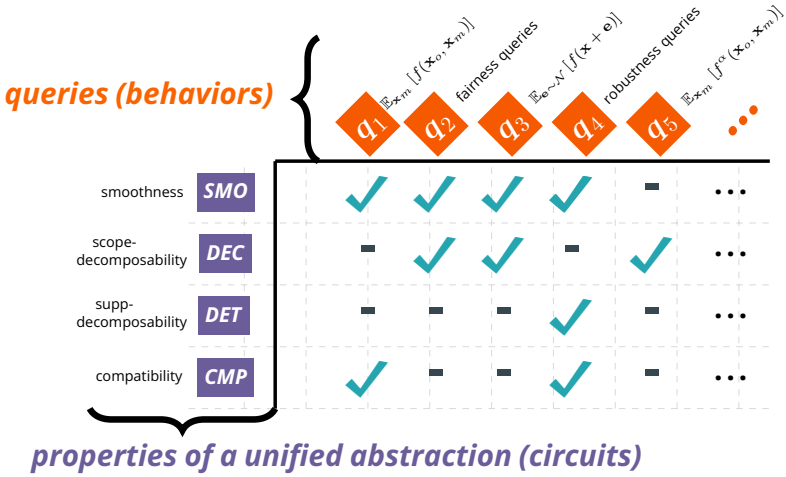
???

```
def var(p, f):
    r = pow(f, 2)
    t = mult(r, p)
    return integrate(t)
```

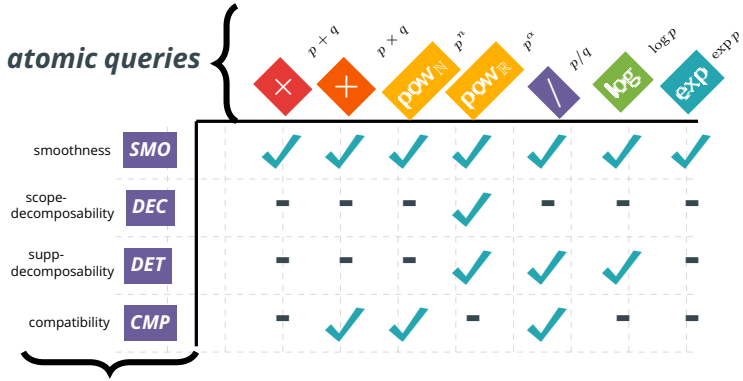
efficient & reliable complex reasoning routines
(fast routines to inspect and guarantee a ML system's behavior)

How can we solve this *engineering bottleneck*?


queries (behaviors)
$\mathbf{q}_1$ $\mathbb{E}_{\mathbf{x}_m}[f(\mathbf{x}_o, \mathbf{x}_m)]$
$\mathbf{q}_2$ fairness queries
$\mathbf{q}_3$ $\mathbb{E}_{\mathbf{e}\sim\mathcal{N}}[f(\mathbf{x}+\mathbf{e})]$
$\mathbf{q}_4$ robustness queries
$\mathbf{q}_5$ $\mathbb{E}_{\mathbf{x}_m}[f^{a_1}(\mathbf{x}_o, \mathbf{x}_m)]$
HMMs M_1
decision trees M_2
neural nets w/ ReLUs M_3
tensor factorizations M_4
transformers M_5
model classes
reliable & efficient algo
intractable
do not know



necess. conditions for reliability and efficiency



✓ necess. conditions for reliability and efficiency

(De)composing queries

*"What is the **expected prediction** for a patient with unavailable records?"*

$$\int \left[p(\mathbf{x}_m \mid \mathbf{x}_o) \times f(\mathbf{x}_o, \mathbf{x}_m) \right]$$

(De)composing queries

*"What is the **expected prediction** for a patient with unavailable records?"*

$$\int \left[p(\mathbf{x}_m \mid \mathbf{x}_o) \times f(\mathbf{x}_o, \mathbf{x}_m) \right]$$

*"How **fair** is the prediction is a certain protected attribute changes?"*

$$\int \left[p(\mathbf{x}_m \mid \mathbf{x}_o) \times \text{pow}(f(\mathbf{x}_o, \mathbf{x}_m), 2) \right] - \text{pow}\left(\int \left[p(\mathbf{x}_m \mid \mathbf{x}_o) \times f(\mathbf{x}_o, \mathbf{x}_m) \right], 2 \right)$$

(De)composing queries

*"What is the **expected prediction** for a patient with unavailable records?"*

$$\int \left[p(\mathbf{x}_m \mid \mathbf{x}_o) \times f(\mathbf{x}_o, \mathbf{x}_m) \right]$$

*"How **fair** is the prediction is a certain protected attribute changes?"*

$$\int \left[p(\mathbf{x}_m \mid \mathbf{x}_o) \times \text{pow}\left(f(\mathbf{x}_o, \mathbf{x}_m), 2\right) \right] - \text{pow}\left(\int \left[p(\mathbf{x}_m \mid \mathbf{x}_o) \times f(\mathbf{x}_o, \mathbf{x}_m) \right], 2 \right)$$

*"Can we certify no **adversarial examples** exist?"*

$$\int \left[p(\mathbf{x}_c \mid X_s = 0) \times f(\mathbf{x}_c, 0) \right] - \int \left[p(\mathbf{x}_c \mid X_s = 1) \times f(\mathbf{x}_c, 1) \right]$$

A language for queries

Integral expressions that can be formed by composing these operators

$+$, $\times$, **pow**, **log**, **exp** and $/$

$\Rightarrow$ *many divergences and information-theoretic queries*

A language for queries

Integral expressions that can be formed by composing these operators

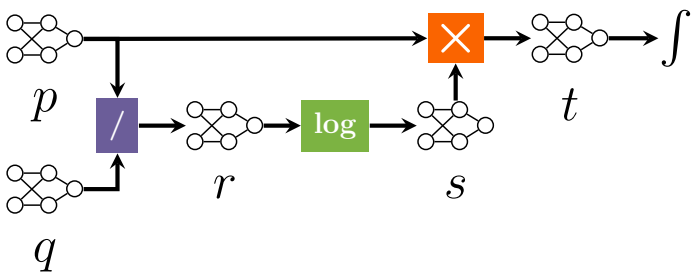
$+$, $\times$, **pow**, **log**, **exp** and $/$

$\Rightarrow$ *many divergences and information-theoretic queries*

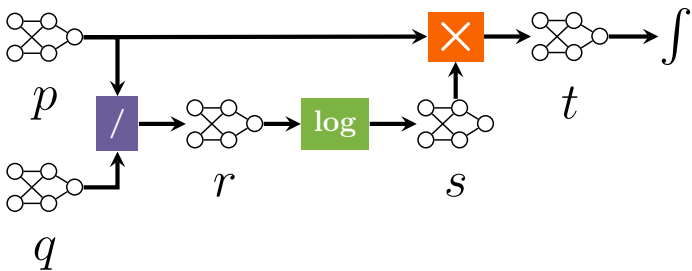
Represented as ***higher-order computational graphs***—pipelines—operating over circuits!

$\Rightarrow$ *re-using intermediate transformations across queries*

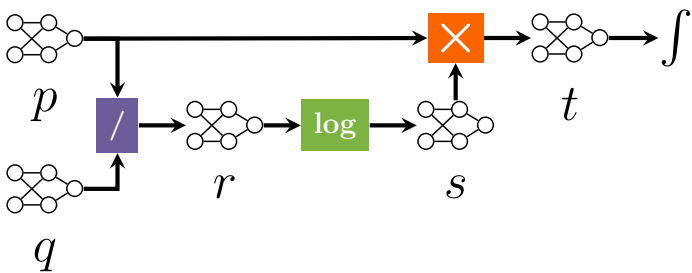
$$\mathbb{KLD}(p \mid\mid q) = \int_{\mathsf{val}(\mathbf{X})} p(\mathbf{x}) \times \log\left(p(\mathbf{x})/q(\mathbf{x})\right)\ d\mathbf{X}$$



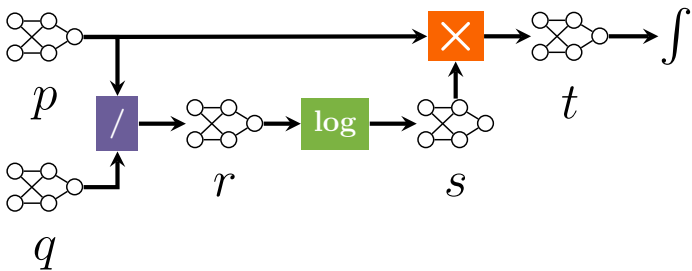
$$\mathbb{KLD}(p \mid\mid q) = \int_{\mathsf{val}(\mathbf{X})} p(\mathbf{x}) \times \log\left(p(\mathbf{x}) \,/\, q(\mathbf{x})\right)\, d\mathbf{X}$$



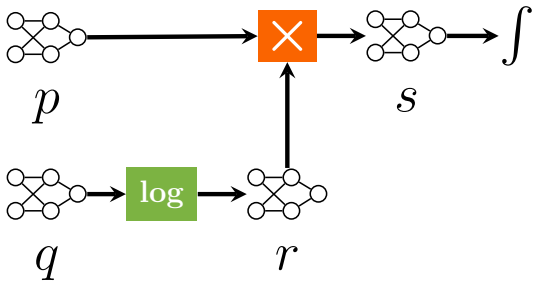
$$\mathbb{KLD}(p \mid\mid q) = \int_{\mathsf{val}(\mathbf{X})} p(\mathbf{x}) \times \log\left(p(\mathbf{x})/q(\mathbf{x})\right)\, d\mathbf{X}$$



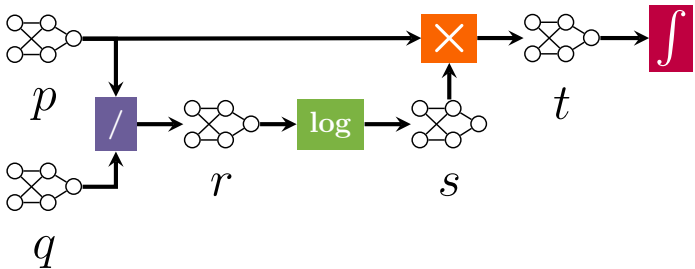
$$\mathbb{KLD}(p \mid\mid q) = \int_{\mathsf{val}(\mathbf{X})} p(\mathbf{x}) \times \log\left(p(\mathbf{x})/q(\mathbf{x})\right)\ d\mathbf{X}$$



$$\mathbb{XENT}(p \mid\mid q) = \int p(\mathbf{x}) \times \log q(\mathbf{x})\, d\mathbf{X}$$

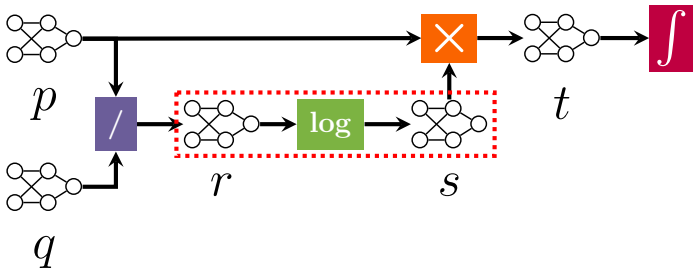


$$\int p(\mathbf{x}) \times \log \left(p(\mathbf{x}) \,/\, q(\mathbf{x}) \right) \, d\mathbf{X}$$



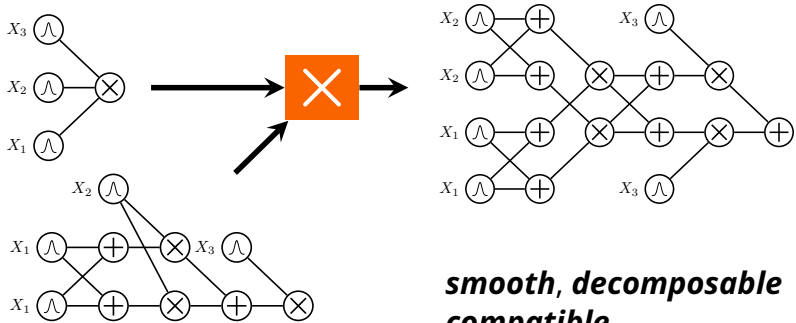
build a LEGO-like query calculus...

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



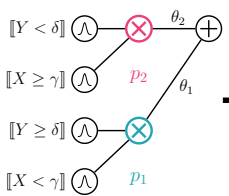
build a LEGO-like query calculus...

Tractable operators

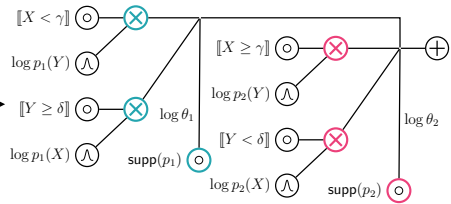


smooth*, *decomposable
compatible

Tractable operators

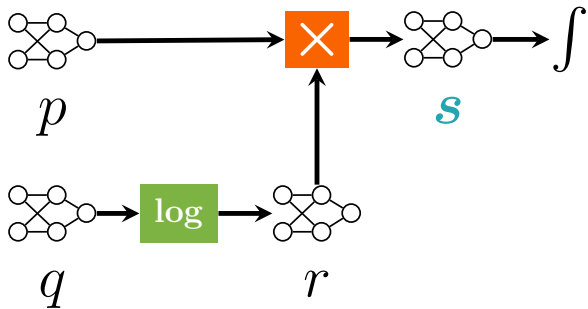


log

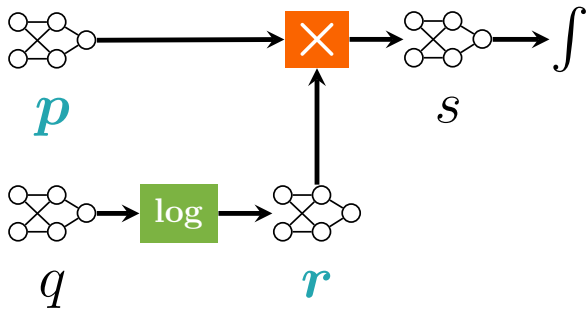


smooth, decomposable deterministic

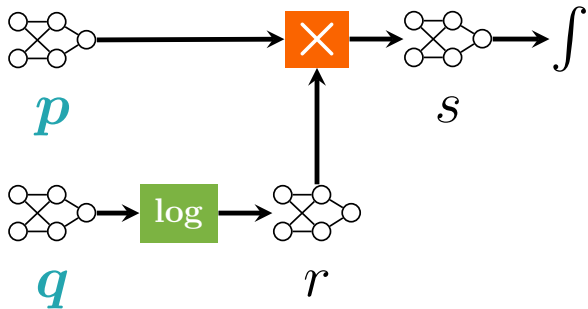
smooth, decomposable



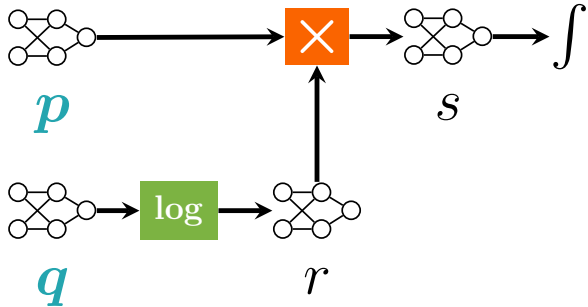
To perform tractable integration we need s to be ***smooth and decomposable...***



hence we need p and r to be smooth, decomposable and ***compatible***...



therefore q must be smooth, decomposable and ***deterministic***...



we can compute $\mathbb{XENT}$ tractably if $\boldsymbol{p}$ and $\boldsymbol{q}$ are smooth, decomposable, compatible and $\boldsymbol{q}$ is deterministic...

	Query	Tract. Conditions	Hardness
CROSS ENTROPY	$-\int p(\boldsymbol{x}) \log q(\boldsymbol{x}) \, \mathrm{d}\mathbf{X}$	Cmp, q Det	#P-hard w/o Det
SHANNON ENTROPY	$-\sum p(\boldsymbol{x}) \log p(\boldsymbol{x})$	Sm, Dec, Det	coNP-hard w/o Det
RÉNYI ENTROPY	$(1-\alpha)^{-1} \log \int p^{\alpha}(\boldsymbol{x}) \, d\mathbf{X}, \alpha \in \mathbb{N}$	SD	#P-hard w/o SD
	$(1-\alpha)^{-1} \log \int p^{\alpha}(\boldsymbol{x}) \, d\mathbf{X}, \alpha \in \mathbb{R}_{+}$	Sm, Dec, Det	#P-hard w/o Det
MUTUAL INFORMATION	$\int p(\boldsymbol{x}, \boldsymbol{y}) \log(p(\boldsymbol{x}, \boldsymbol{y}) / (p(\boldsymbol{x}) p(\boldsymbol{y})))$	Sm, SD, Det*	coNP-hard w/o SD
KULLBACK-LEIBLER DIV.	$\int p(\boldsymbol{x}) \log(p(\boldsymbol{x}) / q(\boldsymbol{x})) d\mathbf{X}$	Cmp, Det	#P-hard w/o Det
RÉNYI'S ALPHA DIV.	$(1-\alpha)^{-1} \log \int p^{\alpha}(\boldsymbol{x}) q^{1-\alpha}(\boldsymbol{x}) \, d\mathbf{X}, \alpha \in \mathbb{N}$	Cmp, q Det	#P-hard w/o Det
	$(1-\alpha)^{-1} \log \int p^{\alpha}(\boldsymbol{x}) q^{1-\alpha}(\boldsymbol{x}) \, d\mathbf{X}, \alpha \in \mathbb{R}$	Cmp, Det	#P-hard w/o Det
ITAKURA-SAITO DIV.	$\int [p(\boldsymbol{x}) / q(\boldsymbol{x}) - \log(p(\boldsymbol{x}) / q(\boldsymbol{x})) - 1] d\,\mathbf{X}$	Cmp, Det	#P-hard w/o Det
CAUCHY-SCHWARZ DIV.	$-\log \frac{\int p(\boldsymbol{x}) q(\boldsymbol{x}) d\mathbf{X}}{\sqrt{\int p^2(\boldsymbol{x}) d\mathbf{X} \int q^2(\boldsymbol{x}) d\mathbf{X}}}$	Cmp	#P-hard w/o Cmp
SQUARED LOSS	$\int (p(\boldsymbol{x}) - q(\boldsymbol{x}))^2 d\,\mathbf{X}$	Cmp	#P-hard w/o Cmp

compositionally *derive the tractability of many more queries*

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

Query		Tract. Conditions	Hardness
CROSS ENTROPY	$-\int p(\boldsymbol{x}) \log q(\boldsymbol{x}) \, \mathrm{d}\mathbf{X}$	Cmp, q Det	#P-hard w/o Det
SHANNON ENTROPY	$-\sum p(\boldsymbol{x}) \log p(\boldsymbol{x})$	Sm, Dec, Det	coNP-hard w/o Det
RÉNYI ENTROPY	$(1-\alpha)^{-1} \log \int p^{\alpha}(\boldsymbol{x}) \, d\mathbf{X}, \alpha \in \mathbb{N}$	SD	#P-hard w/o SD
	$(1-\alpha)^{-1} \log \int p^{\alpha}(\boldsymbol{x}) \, d\mathbf{X}, \alpha \in \mathbb{R}_{+}$	Sm, Dec, Det	#P-hard w/o Det
MUTUAL INFORMATION	$\int p(\boldsymbol{x}, \boldsymbol{y}) \log(p(\boldsymbol{x}, \boldsymbol{y}) / (p(\boldsymbol{x}) p(\boldsymbol{y})))$	Sm, SD, Det*	coNP-hard w/o SD
KULLBACK-LEIBLER DIV.	$\int p(\boldsymbol{x}) \log(p(\boldsymbol{x}) / q(\boldsymbol{x})) d\mathbf{X}$	Cmp, Det	#P-hard w/o Det
RÉNYI'S ALPHA DIV.	$(1-\alpha)^{-1} \log \int p^{\alpha}(\boldsymbol{x}) q^{1-\alpha}(\boldsymbol{x}) \, d\mathbf{X}, \alpha \in \mathbb{N}$	Cmp, q Det	#P-hard w/o Det
	$(1-\alpha)^{-1} \log \int p^{\alpha}(\boldsymbol{x}) q^{1-\alpha}(\boldsymbol{x}) \, d\mathbf{X}, \alpha \in \mathbb{R}$	Cmp, Det	#P-hard w/o Det
ITAKURA-SAITO DIV.	$\int [p(\boldsymbol{x}) / q(\boldsymbol{x}) - \log(p(\boldsymbol{x}) / q(\boldsymbol{x})) - 1] d\,\mathbf{X}$	Cmp, Det	#P-hard w/o Det
CAUCHY-SCHWARZ DIV.	$-\log \frac{\int p(\boldsymbol{x}) q(\boldsymbol{x}) d\mathbf{X}}{\sqrt{\int p^2(\boldsymbol{x}) d\mathbf{X} \int q^2(\boldsymbol{x}) d\mathbf{X}}}$	Cmp	#P-hard w/o Cmp
SQUARED LOSS	$\int (p(\boldsymbol{x}) - q(\boldsymbol{x}))^2 d\,\mathbf{X}$	Cmp	#P-hard w/o Cmp

*and **prove hardness** when some input properties are not satisfied*

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

Composable tractable sub-routines

```
function kld(p, q)
    r = quotient(p, q)
    s = log(r)
    t = product(p, s)
    return integrate(t)
end
```

```
function xent(p, q)
    r = log(q)
    s = product(p, r)
    return -integrate(s)
end
```

```
function ent(p)
    q = log(p)
    r = product(p, q)
    return -integrate(s)
end
```

```
function alphadiv(p, q, alpha=1.5)
    r = real_pow(p, alpha)
    s = real_pow(q, 1.0-alpha)
    t = product(r, s)
    return log(integrate(t)) / (1.0-alpha)
end
```

Efficient inference algorithms ***in a couple lines of Julia code!*** [1]

[1] https://github.com/UCLA-StarAI/circuit-ops-atlas