Tractable Probabilistic Modeling with Probabilistic Circuits

antonio vergari (he/him)

@tetraduzione

20th June 2024 - **Nordic Prob AI Summer School** - *Copenhagen*

april

april is probably a recursive identifier of a lab

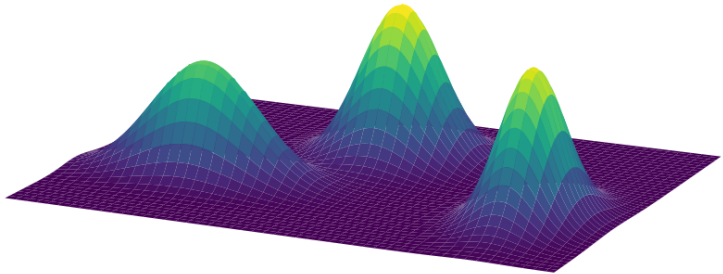
april

about
probabilities
integrals &
logic

deep generative models

+

flexible and reliable
logic & probabilistic reasoning?



a love letter to mixture models...

image taken from Hao Tang's course on ASR

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

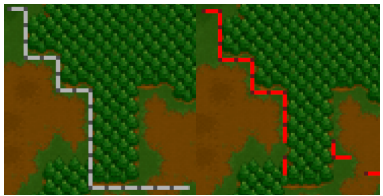
q1 $\int p(\mathbf{x}_o, \mathbf{x}_m) d\mathbf{X}_m$
(missing values)

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.)

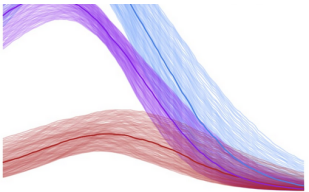
...in the language of probabilities

more complex reasoning



neuro-symbolic AI

probabilistic programming



computing uncertainties (Bayesian inference)

...and more application scenarios

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.)

hard to compute in general!

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!

Tractable Probabilistic Inference

A class of queries $\mathcal{Q}$ *is tractable on a family of probabilistic models* $\mathcal{M}$

iff for any query $\mathbf{q} \in \mathcal{Q}$ *and model* $\mathbf{m} \in \mathcal{M}$

exactly *computing* $\mathbf{q}(\mathbf{m})$ *runs in time* $O(\text{poly}(|\mathbf{m}|))$.

$\Rightarrow$ ***model-centric*** *definition...*

Tractable Probabilistic Inference

A class of queries $\mathcal{Q}$ *is tractable on a family of probabilistic models* $\mathcal{M}$

iff for any query $\mathbf{q} \in \mathcal{Q}$ *and model* $\mathbf{m} \in \mathcal{M}$

exactly *computing* $\mathbf{q}(\mathbf{m})$ *runs in time* $O(\text{poly}(|\mathbf{m}|))$.

$\Rightarrow$ ***model-centric*** *definition...*

$\Rightarrow$ *...and* ***query-centric****: Tractability is not a universal property!*

Tractable Probabilistic Inference

A class of queries $\mathcal{Q}$ *is tractable on a family of probabilistic models* $\mathcal{M}$

iff for any query $\mathbf{q} \in \mathcal{Q}$ *and model* $\mathbf{m} \in \mathcal{M}$

exactly *computing* $\mathbf{q}(\mathbf{m})$ *runs in time* $O(\text{poly}(|\mathbf{m}|))$.

$\Rightarrow$ ***model-centric*** *definition...*

$\Rightarrow$ *...and* ***query-centric****: Tractability is not a universal property!*

$\Rightarrow$ *often* poly *will in fact be* ***linear****!*

Tractable Probabilistic Inference

A class of queries $\mathcal{Q}$ *is tractable on a family of probabilistic models* $\mathcal{M}$

iff for any query $\mathbf{q} \in \mathcal{Q}$ *and model* $\mathbf{m} \in \mathcal{M}$

exactly *computing* $\mathbf{q}(\mathbf{m})$ *runs in time* $O(\text{poly}(|\mathbf{m}|))$.

$\Rightarrow$ ***model-centric*** *definition...*

$\Rightarrow$ *...and* ***query-centric****: Tractability is not a universal property!*

$\Rightarrow$ *often* poly *will in fact be* ***linear****!*

$\Rightarrow$ *Note: if* $|\mathbf{m}| \in O(\text{poly}(|\mathbf{X}|))$*, then query time is* $O(\text{poly}(|\mathbf{X}|))$.

Tractable Probabilistic Inference

A class of queries $\mathcal{Q}$ *is tractable on a family of probabilistic models* $\mathcal{M}$

iff for any query $\mathbf{q} \in \mathcal{Q}$ *and model* $\mathbf{m} \in \mathcal{M}$

exactly *computing* $\mathbf{q}(\mathbf{m})$ *runs in time* $O(\text{poly}(|\mathbf{m}|))$.

$\Rightarrow$ ***model-centric*** *definition...*

$\Rightarrow$ *...and* ***query-centric****: Tractability is not a universal property!*

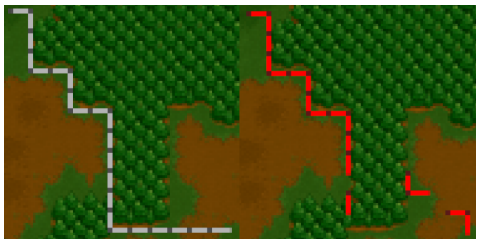
$\Rightarrow$ *often* poly *will in fact be* ***linear****!*

$\Rightarrow$ *Note: if* $|\mathbf{m}| \in O(\text{poly}(|\mathbf{X}|))$*, then query time is* $O(\text{poly}(|\mathbf{X}|))$.

$\Rightarrow$ *Why* ***exactness****? Highest guarantee possible!*

why tractable models?

exactness can be crucial in safety-driven applications



guarantee constraint satisfaction

[Ahmed et al. 2022]

estimation error is bounded (0)

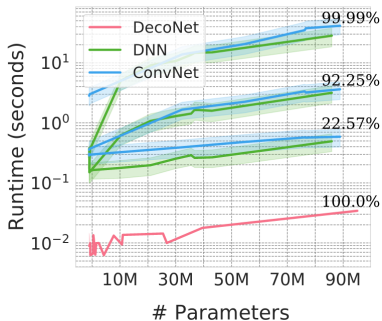
[Choi 2022]

why tractable models?

they can be much faster than intractable ones!

Method	MNIST (10,000 test images)		
	Theoretical bpd	Comp. bpd	En- & decoding time
PC (small)	1.26	1.30	**53**
PC (large)	**1.20**	**1.24**	168
IDF	1.90	1.96	880
BitSwap	1.27	1.31	904

[Liu, Mandt, and Broeck 2022]



[Subramani et al. 2021]

Why?

tractable inference

Why?

we always perform
tractable inference
over an approximate model!

VI

$$\min_{q \in \mathcal{Q}} \mathrm{KL}(q||p)$$

we pick a ***tractable*** variational distribution q

$\Rightarrow$ *e.g., Gaussian, GMM, HMM, flow, etc*

VI

$$\min_{q \in \mathcal{Q}} \mathrm{KL}(q||p)$$

we pick a ***tractable*** variational distribution q

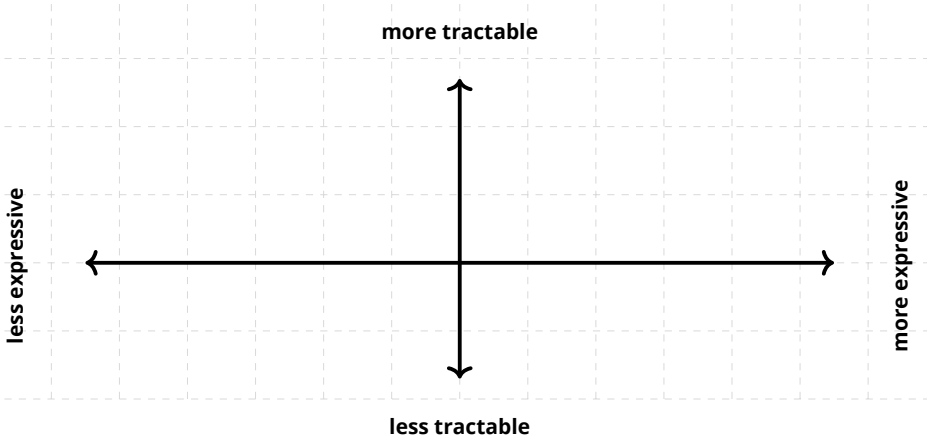
MC

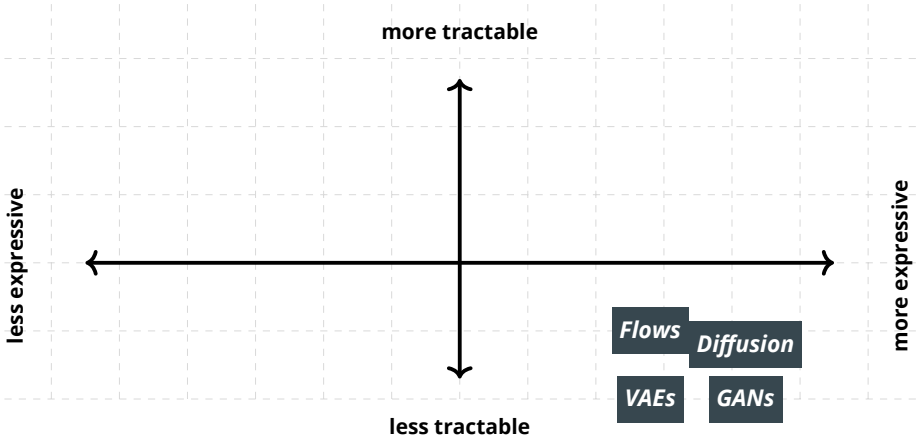
$$\mathbb{E}_{p(\mathbf{x})}[f(\mathbf{x})] \approx \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}^{(i)})$$

we turn an intractable integral into a ***tractable sum***

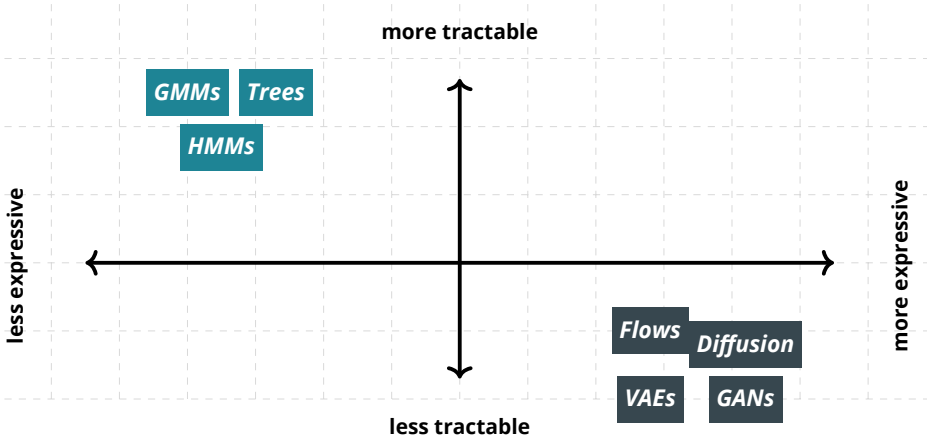
Goal

"Can we find
a middle ground
between
tractability and expressiveness?"

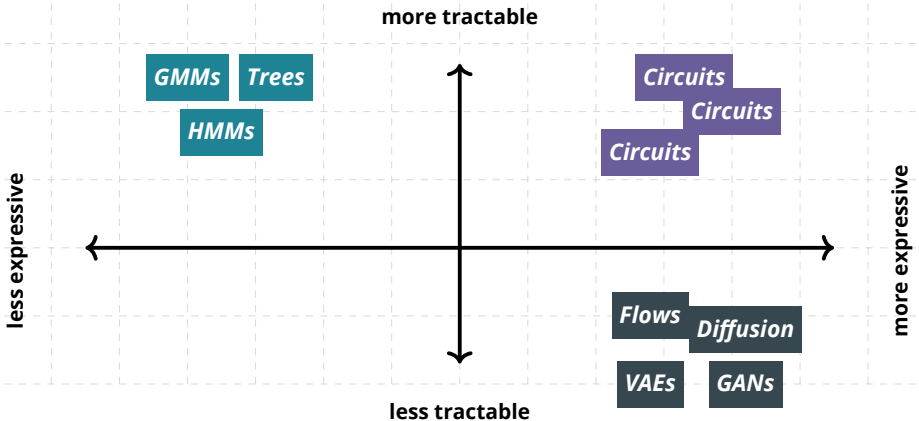

more tractable
less expressive
more expressive
less tractable



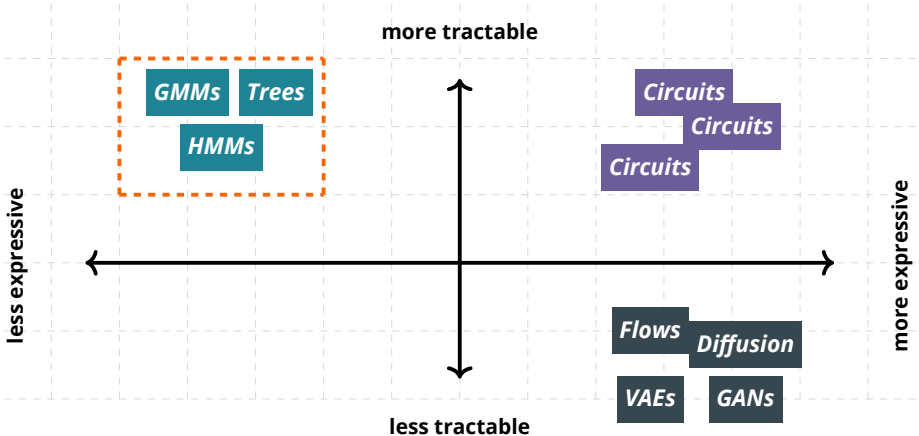
expressive models are not much tractable...



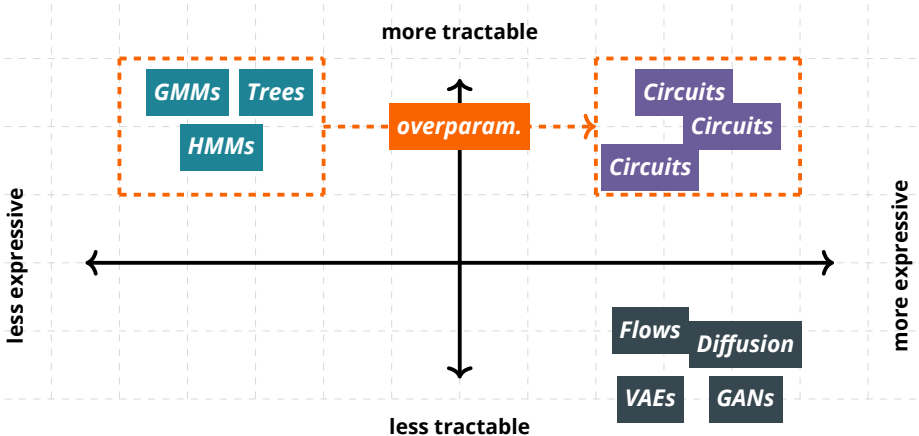
tractable models are not that expressive...



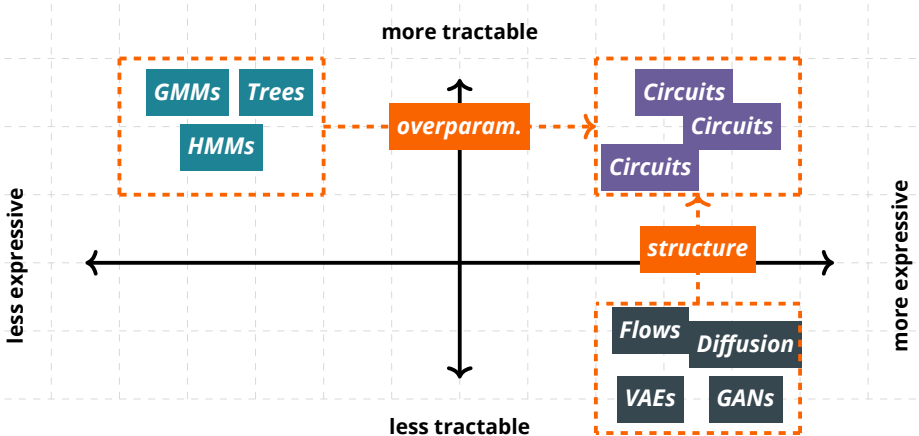
circuits can be both expressive and tractable!



start simple...



then make it more expressive!



impose structure!

Goal

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

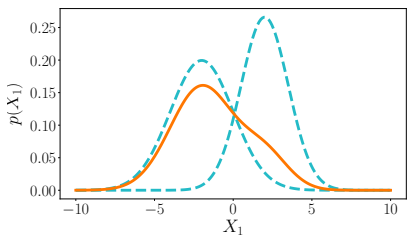
Goal

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

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

GMMs

as computational graphs

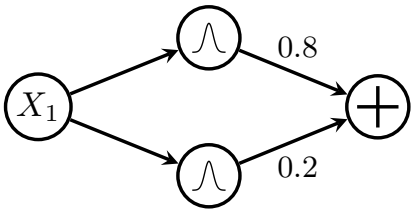


$$p(X) = w_1 \cdot p_1(X_1) + w_2 \cdot p_2(X_1)$$

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

GMMs

as computational graphs

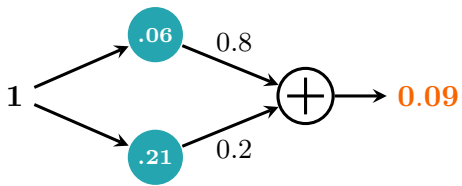


$$p(X_1) = 0.2 \cdot p_1(X_1) + 0.8 \cdot p_2(X_1)$$

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

GMMs

as computational graphs

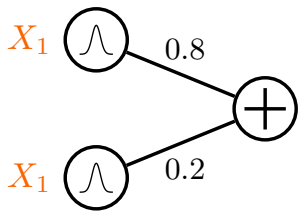


$$p(X = 1) = 0.2 \cdot p_1(X_1 = 1) + 0.8 \cdot p_2(X_1 = 1)$$

$\Rightarrow$ *inference = feedforward evaluation*

GMMs

as computational graphs



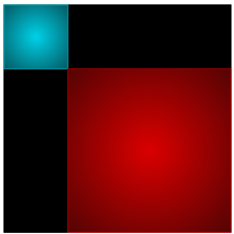
A simplified notation:

⇒ ***scopes*** *attached to inputs*

⇒ *edge directions omitted*

GMMs

as computational graphs

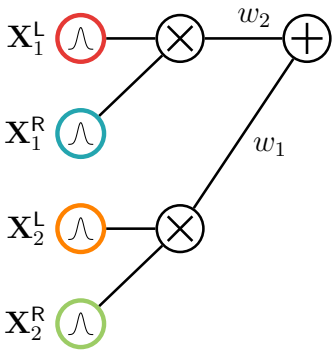


$$p(\mathbf{X}) = w_1 \cdot p_1(\mathbf{X}_1^{\mathsf{L}}) \cdot p_1(\mathbf{X}_1^{\mathsf{R}}) + w_2 \cdot p_2(\mathbf{X}_2^{\mathsf{L}}) \cdot p_2(\mathbf{X}_2^{\mathsf{R}})$$

$\Rightarrow$ *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

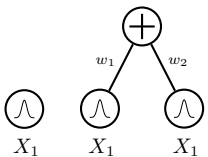
I. *A simple tractable function is a circuit*

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*



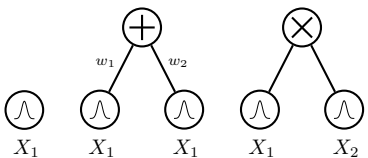
Probabilistic Circuits (PCs)

A grammar for tractable computational graphs

I. *A simple tractable function is a circuit*

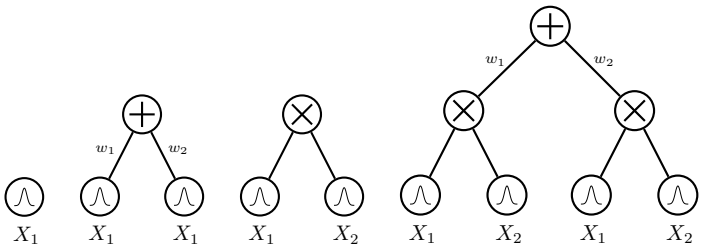
II. *A weighted combination of circuits is a circuit*

III. *A product of circuits is a circuit*



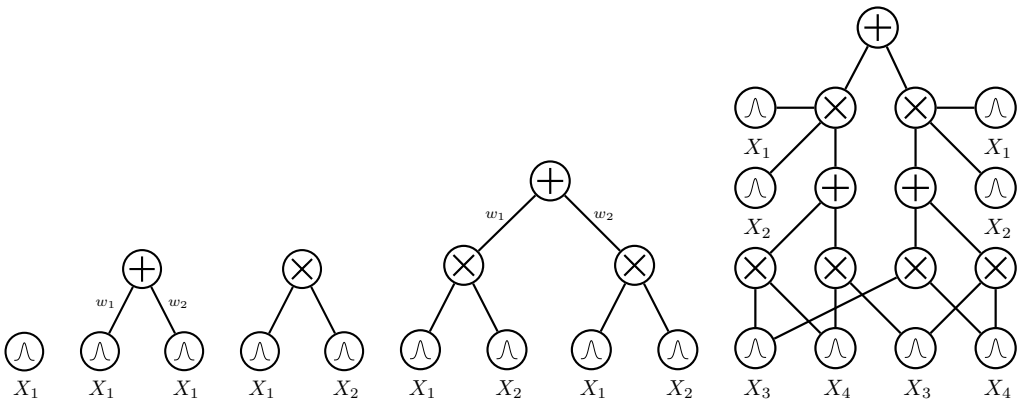
Probabilistic Circuits (PCs)

A grammar for tractable computational graphs



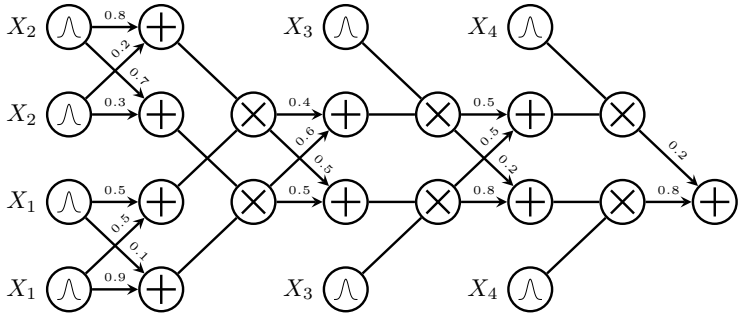
Probabilistic Circuits (PCs)

A grammar for tractable computational graphs



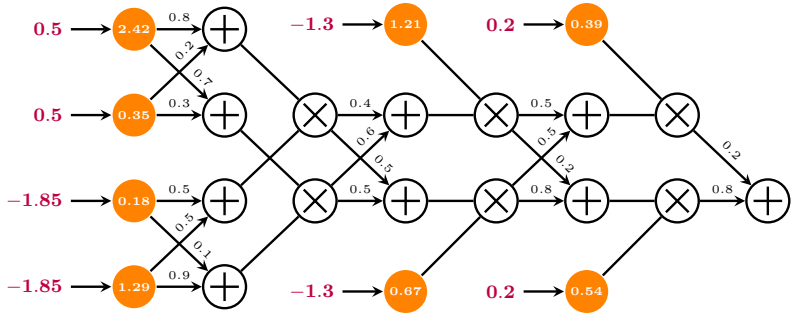
Probabilistic queries = **feedforward** evaluation

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



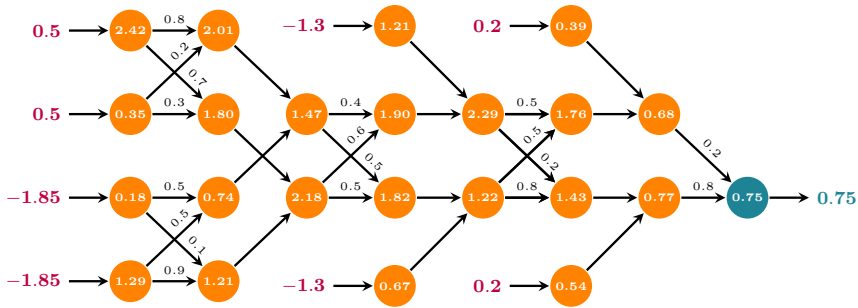
Probabilistic queries = **feedforward** evaluation

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



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

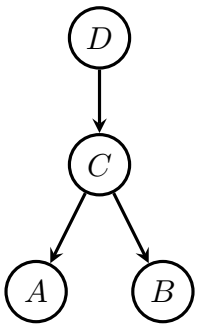
One formalism to represent many probabilistic and logical models

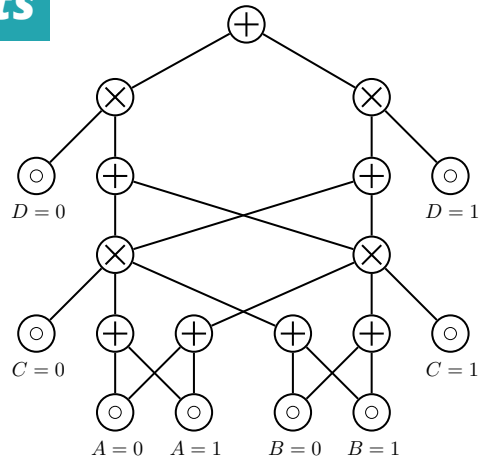
$\Rightarrow$ *#HMMs #Trees #XGBoost, Tensor Networks, ...*

and other *PGMs*...

From PGMs to circuits

via compilation

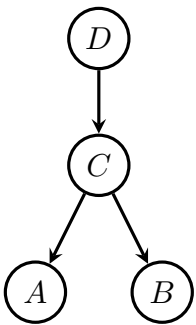




From PGMs to circuits

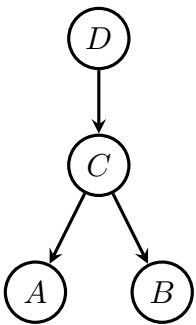
via compilation

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



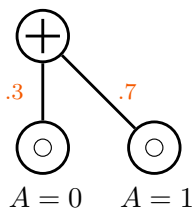
From PGMs to circuits

via compilation



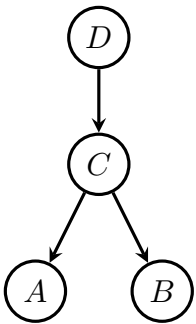
...compile a leaf CPT

$p(A|C=0)$



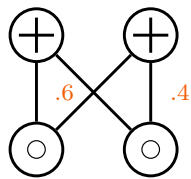
From PGMs to circuits

via compilation



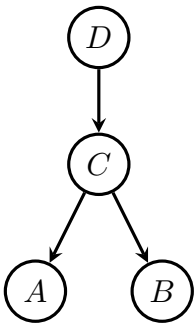
...compile a leaf CPT

$p(A|C=1)$

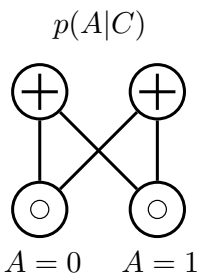


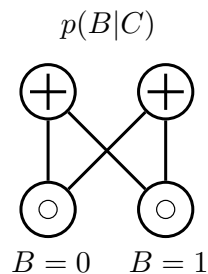
From PGMs to circuits

via compilation



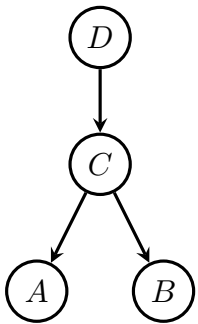
...compile a leaf CPT...for all leaves...



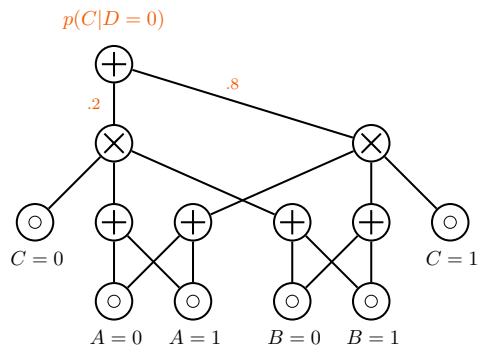


From PGMs to circuits

via compilation

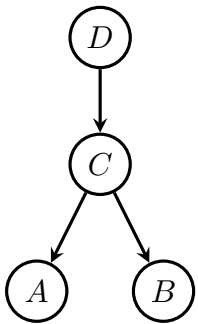


...and recurse over parents...

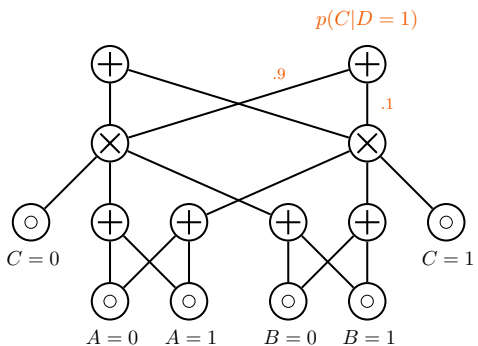


From PGMs to circuits

via compilation

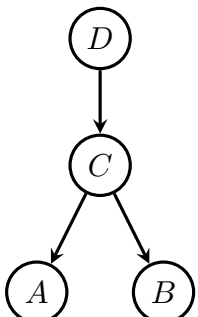


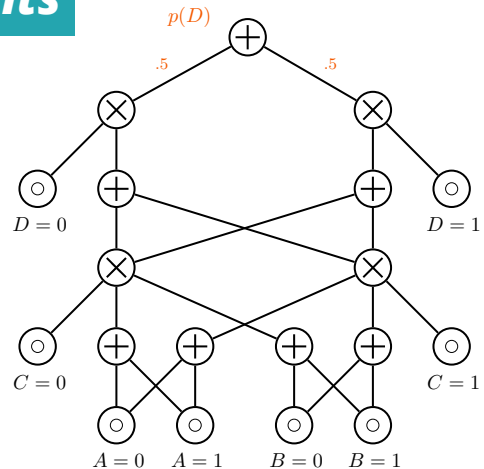
...while reusing previously compiled nodes!...



From PGMs to circuits

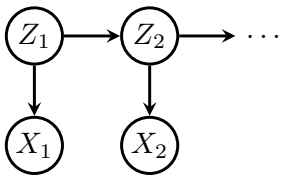
via compilation

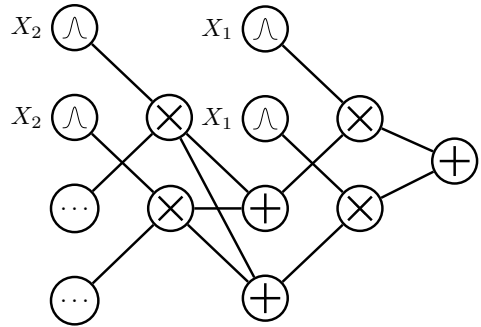




HMMs

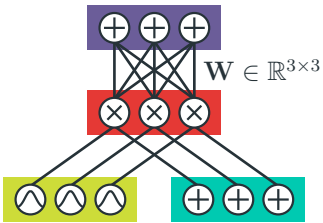
as computational graphs

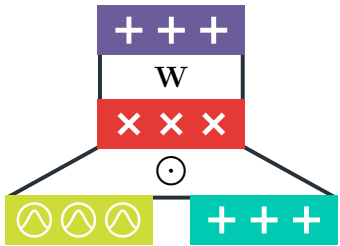




overparameterize

and tensorize

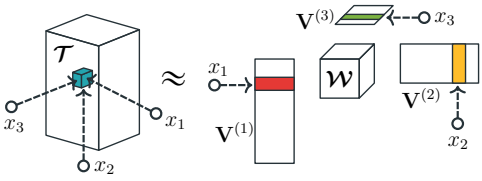


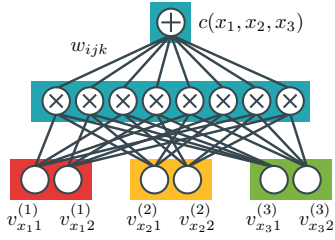


Mari, Vessio, and Vergari, "Unifying and Understanding Overparameterized Circuit Representations via Low-Rank Tensor Decompositions", 6th Workshop on Tractable Probabilistic Modeling, 2023

tensor factorizations

as circuits





Mari et al., "What is the Relationship between Circuits and Tensor Factorizations (and How Can We Exploit it)?", Under submission, 2024

...wait!

“Are all PGMs circuits?”

and/or

“Are all circuits PGMs?”

...not so easy!

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

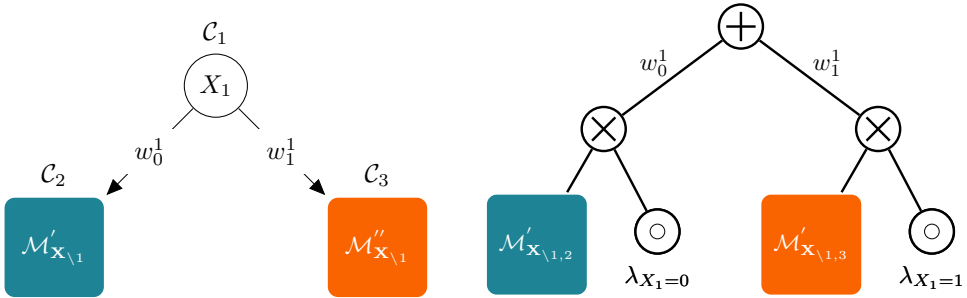
but PCs can exploit context specific independence

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

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]

...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?

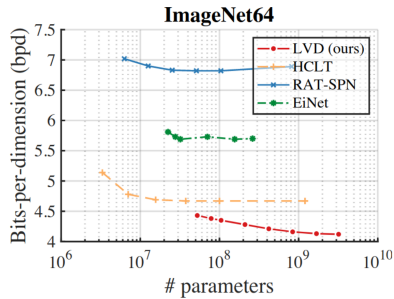
	QPC	PC	Sp-PC	HCLT	RAT	IDF	BitS	BBans	McB
MNIST	**1.11**	1.17	1.14	1.21	1.67	1.90	1.27	1.39	1.98
F-MNIST	**3.16**	3.32	3.27	3.34	4.29	3.47	3.28	3.66	3.72
EMN-MN	1.55	1.64	**1.52**	1.70	2.56	2.07	1.88	2.04	2.19
EMN-LE	**1.54**	1.62	1.58	1.75	2.73	1.95	1.84	2.26	3.12
EMN-BA	**1.59**	1.66	1.60	1.78	2.78	2.15	1.96	2.23	2.88
EMN-BY	1.53	**1.47**	1.54	1.73	2.72	1.98	1.87	2.23	3.14

competitive with Flows and VAEs!

Gala et al., Scaling Continuous Latent Variable Models as Probabilistic Integral Circuits, , 2024

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", The Eleventh International Conference on Learning Representations, 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 for Probabilistic Inference", NeurIPS, 2021*

Structural properties

property A

property B

property C

property D

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

Structural properties

property A

property B

property C

property D

tractable computation of ***arbitrary integrals***

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

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

$\Rightarrow$ *tractable partition function*

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

Structural properties

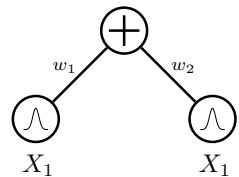
smoothness

decomposability

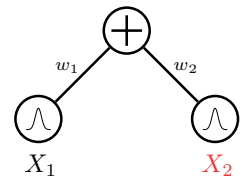
compatibility

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 for Probabilistic Inference", NeurIPS, 2021*

Structural properties

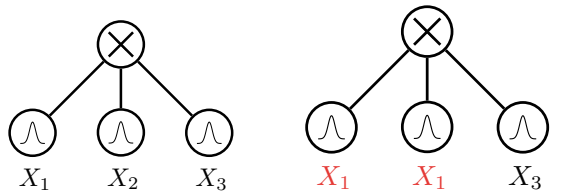
smoothness

decomposability

compatibility

determinism

the inputs of prod units are defined over disjoint variable sets

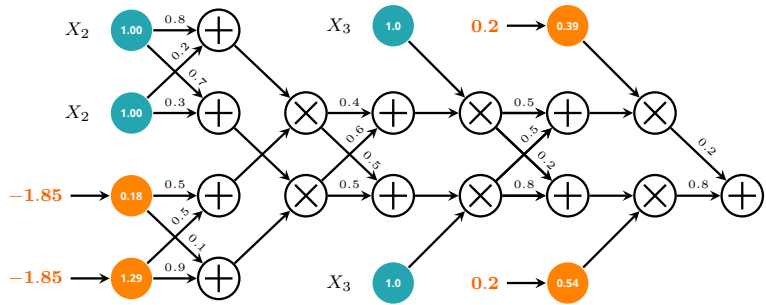


decomposable circuit ***non-decomposable circuit***

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

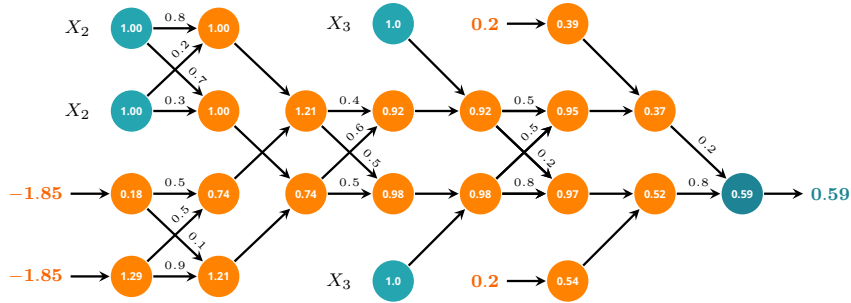
Probabilistic queries = **feedforward** evaluation

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



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:

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

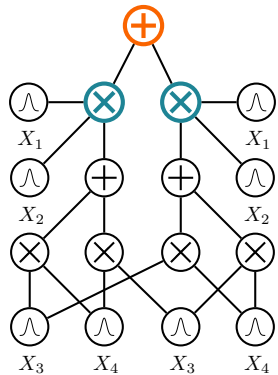
smooth + ***decomposable*** circuits = ...

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*

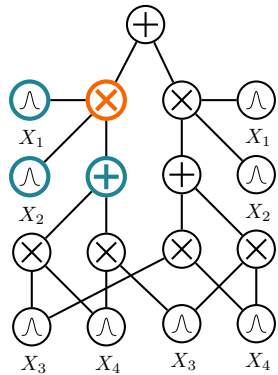


smooth + ***decomposable*** circuits = ...

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*

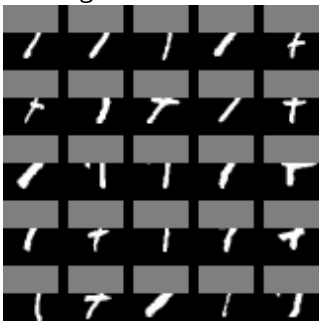


Tractable inference on PCs

Einsum networks

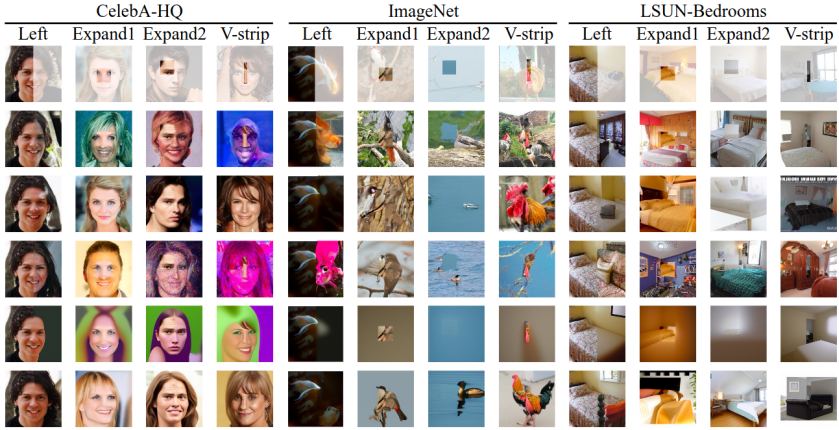
Original

Missing



Conditional sample

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



Liu, Niepert, and Broeck, "Image Inpainting via Tractable Steering of Diffusion Models", ICLR, 2024

Which structural properties

for complex reasoning

smooth + decomposable

Which structural properties

for complex reasoning

smooth + decomposable

??????

Adversarial smoothing

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

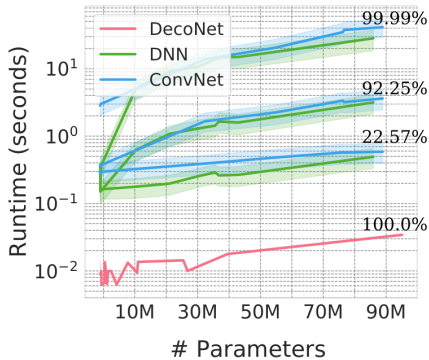
$$g_\sigma(\mathbf{x}) = \mathbb{E}_{\mathbf{e}\sim\mathcal{N}(0,\sigma\mathbf{I})}\left[f(\mathbf{x}+\mathbf{e})\right]$$

Subramani et al., "Exact and Efficient Adversarial Robustness with Decomposable Neural Networks", , 2021

Adversarial smoothing

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

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



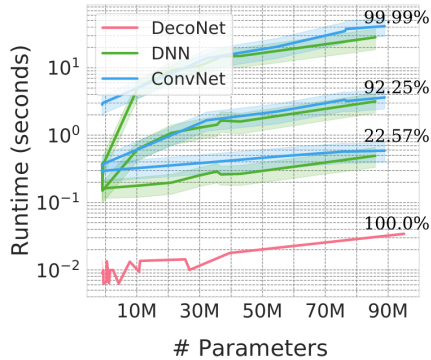
Subramani et al., "Exact and Efficient Adversarial Robustness with Decomposable Neural Networks", , 2021

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



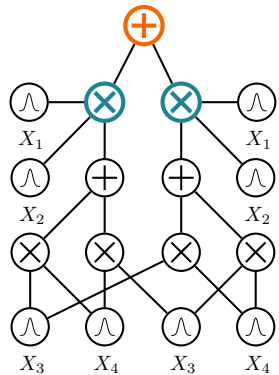
Subramani et al., "Exact and Efficient Adversarial Robustness with Decomposable Neural Networks", , 2021

decomposable circuits = tractable *adv smoothing*

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

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

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



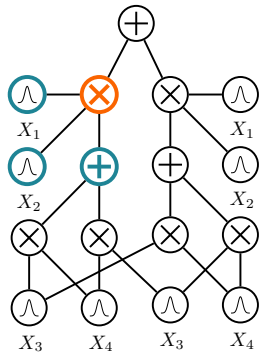
decomposable circuits = tractable *adv smoothing*

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 for Probabilistic Inference", NeurIPS, 2021*

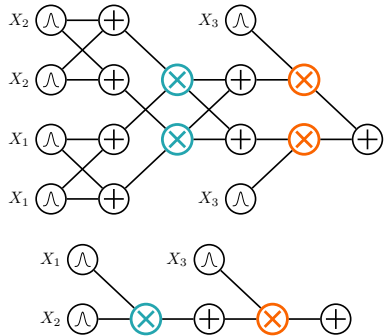
Structural properties

smoothness

decomposability

compatibility

determinism



compatible circuits

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

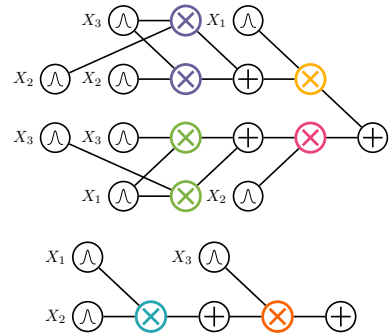
Structural properties

smoothness

decomposability

compatibility

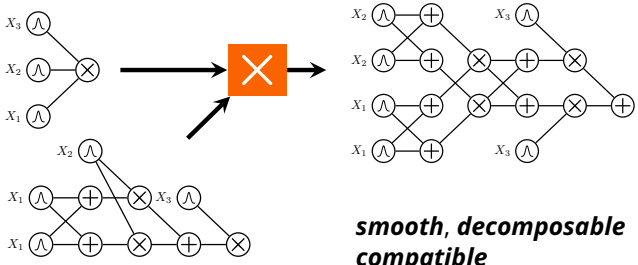
determinism



non-compatible circuits

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

Tractable products



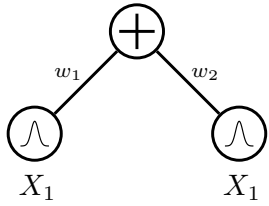
smooth*, *decomposable
compatible

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 for Probabilistic Inference", 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

Structural properties

smoothness

decomposability

compatibility

determinism

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

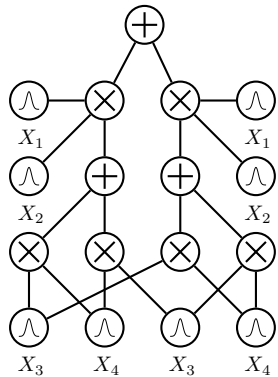
Determinism + **decomposability** = **tractable MAP**

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

$\Rightarrow$ *linear in circuit size!*

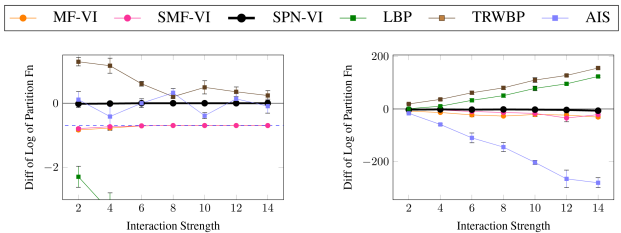
E.g., suppose we want to compute:

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



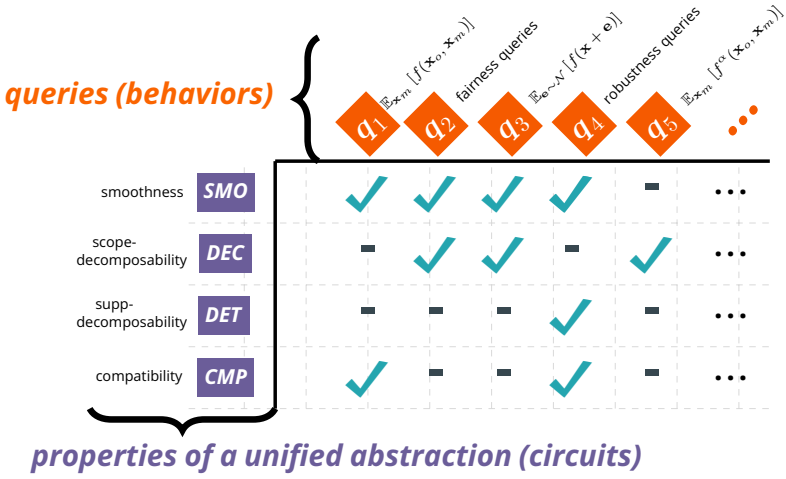
Determinism + **decomposability** = **tractable ELBO**

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

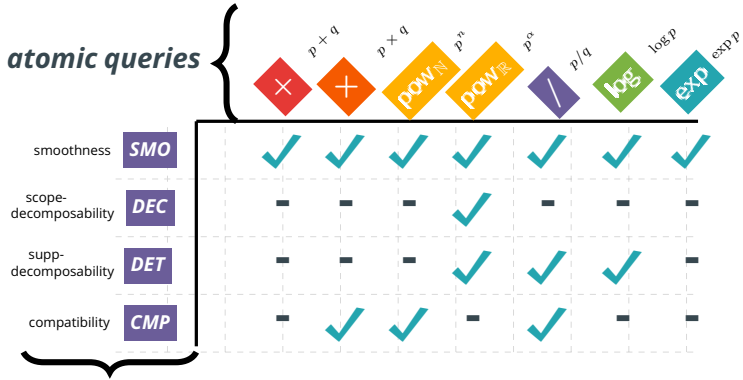


Closed-form computation for the entropy $\mathbb{H}$ *[**Vergari** et al. 2021]*

Shih and Ermon, "Probabilistic Circuits for Variational Inference in Discrete Graphical Models", NeurIPS, 2020



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?"*

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

*"What's the **variance** of my prediction?"*

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

(De)composing queries

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

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

*"What's the **variance** of my prediction?"*

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

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

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

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

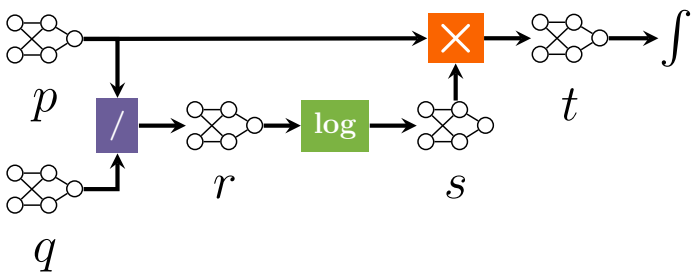
$+$, $\times$, **pow**, **log**, **exp** and $/$

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

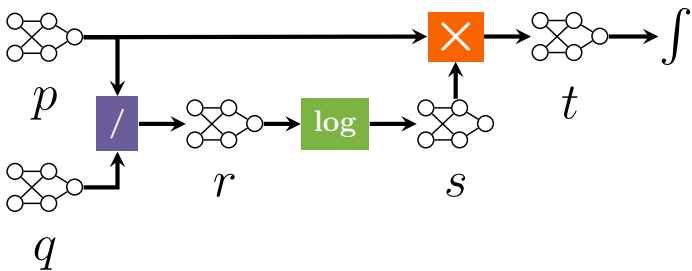
Represented as ***higher-order computational graphs***—pipelines—operating over circuits!

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

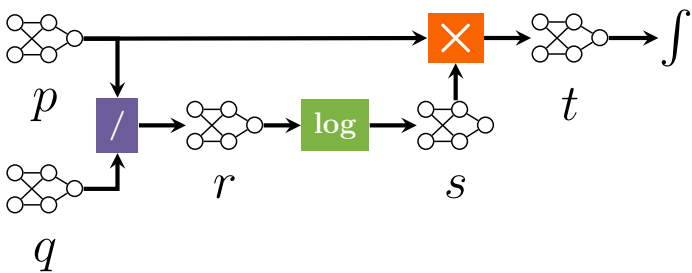
$$\mathbb{KLD}(p \mid\mid q) = \int_{\mathsf{val}(\mathbf{X})} p(\mathbf{x}) \times \log\left(p(\mathbf{x})/q(\mathbf{x})\right)\ d\mathbf{X}$$



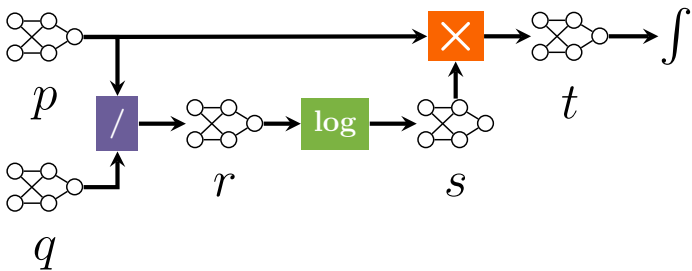
$$\mathbb{KLD}(p \mid\mid q) = \int_{\mathsf{val}(\mathbf{X})} p(\mathbf{x}) \times \log\left(p(\mathbf{x}) \,/\, q(\mathbf{x})\right)\, d\mathbf{X}$$



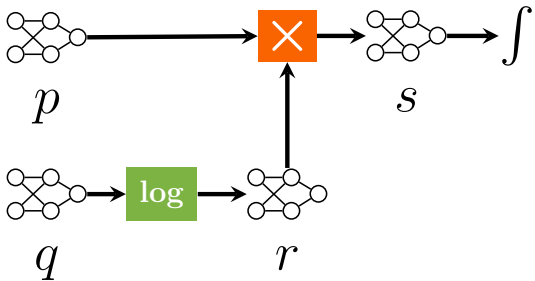
$$\mathbb{KLD}(p \mid\mid q) = \int_{\mathsf{val}(\mathbf{X})} p(\mathbf{x}) \times \log\left(p(\mathbf{x})/q(\mathbf{x})\right)\, d\mathbf{X}$$



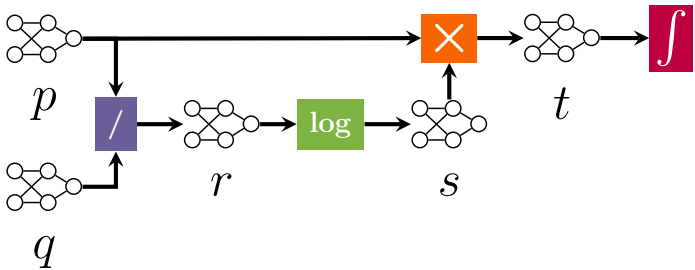
$$\mathbb{KLD}(p \mid\mid q) = \int_{\mathsf{val}(\mathbf{X})} p(\mathbf{x}) \times \log\left(p(\mathbf{x})/q(\mathbf{x})\right)\ d\mathbf{X}$$



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

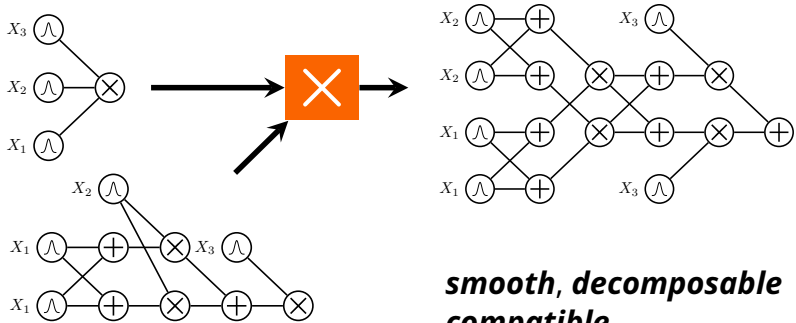


$$\int p(\mathbf{x}) \times \log \left(p(\mathbf{x}) \,/\, q(\mathbf{x}) \right) \, d\mathbf{X}$$



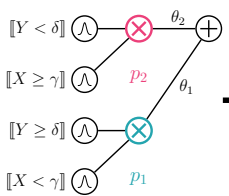
build a LEGO-like query calculus...

Tractable operators

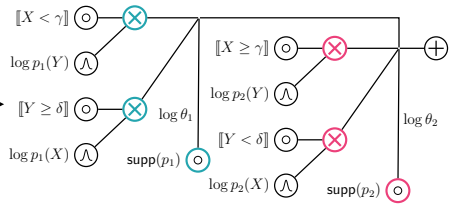


smooth*, *decomposable
compatible

Tractable operators

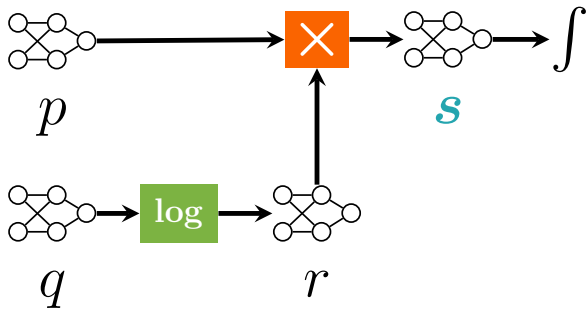


log

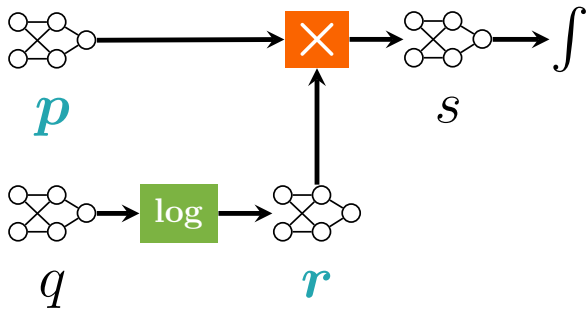


smooth, decomposable deterministic

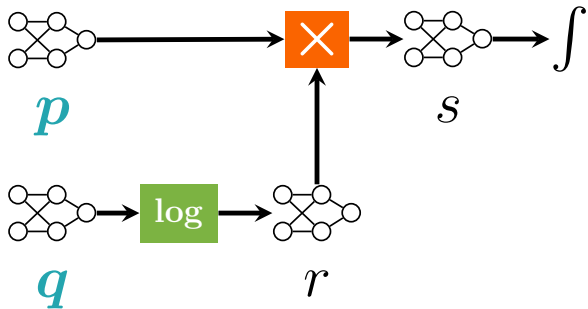
smooth, decomposable



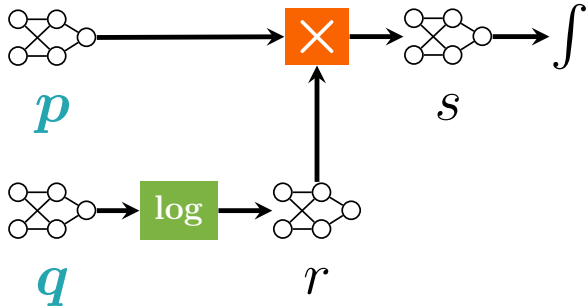
To perform tractable integration we need s to be ***smooth and decomposable...***



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

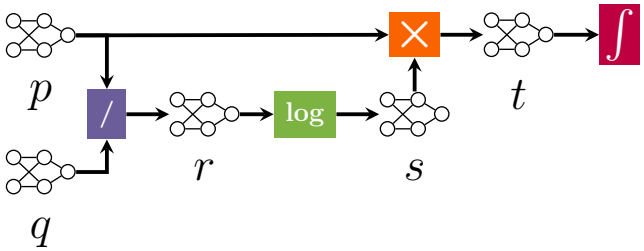


therefore q must be smooth, decomposable and ***deterministic***...



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

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



build a LEGO-like query calculus!

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

compositionally *derive the tractability of many more queries*

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

Learning probabilistic circuits

Learning probabilistic circuits

Probabilistic circuits are (peculiar) neural networks… ***just backprop with SGD!***

Learning probabilistic circuits

Probabilistic circuits are (peculiar) neural networks… ***just backprop with SGD!***

…end of Learning section!

Learning probabilistic circuits

Probabilistic circuits are (peculiar) neural networks... ***just backprop with SGD!***

wait but...

SGD is slow to converge...can we do better?

How to learn normalized weights?

Can we exploit structural properties somehow?

Expectation-Maximization

for smooth and decomposable circuits

$$w_{i,j}^{new} \leftarrow \frac{\sum_{\mathbf{x}\in\mathcal{D}} p[ctx_i = 1, Z_i = j \mid \mathbf{x}; \mathbf{w}^{old}]}{\sum_{\mathbf{x}\in\mathcal{D}} p[ctx_i = 1 \mid \mathbf{x}; \mathbf{w}^{old}]}$$

Darwiche, "A Differential Approach to Inference in Bayesian Networks", , 2003
Peharz et al., "On the Latent Variable Interpretation in Sum-Product Networks", , 2016

Expectation-Maximization

for smooth and decomposable circuits

$$w_{i,j}^{new} \leftarrow \frac{\sum_{\mathbf{x}\in\mathcal{D}} p[ctx_i = 1, Z_i = j \mid \mathbf{x}; \mathbf{w}^{old}]}{\sum_{\mathbf{x}\in\mathcal{D}} p[ctx_i = 1 \mid \mathbf{x}; \mathbf{w}^{old}]}$$

We get **all** the required statistics with a single backprop pass:

$$p[ctx_i = 1, Z_i = j \mid \mathbf{x}; \mathbf{w}^{old}] = \frac{1}{p(\mathbf{x})} \frac{\partial p(\mathbf{x})}{\partial \mathsf{S}_i(\mathbf{x})} \mathsf{N}_j(\mathbf{x}) w_{i,j}^{old}$$

Darwiche, "A Differential Approach to Inference in Bayesian Networks", , 2003
Peharz et al., "On the Latent Variable Interpretation in Sum-Product Networks", , 2016

Bayesian parameter learning

Formulate a prior $p(\mathbf{w}, \boldsymbol{\theta})$ over sum-weights and parameters of input units. Then perform posterior inference:

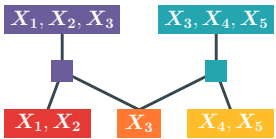
$$p(\mathbf{w}, \boldsymbol{\theta}|\mathcal{D}) \propto p(\mathbf{w}, \boldsymbol{\theta})\, p(\mathcal{D}|\mathbf{w}, \boldsymbol{\theta})$$

Moment matching (oBMM) *[Jaini et al. 2016; Rashwan, Zhao, and Poupart 2016]*

Collapsed variational inference algorithm *[Zhao et al. 2016]*

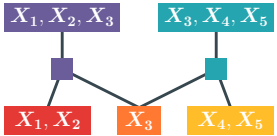
Gibbs sampling *[Trapp et al. 2019; Vergari et al. 2019]*

Learning recipe

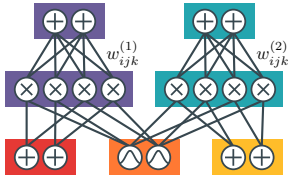


1) Build a ***region graph***

Learning recipe

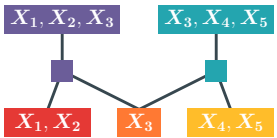


1) Build a ***region graph***

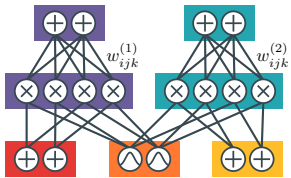


2) Overparameterize

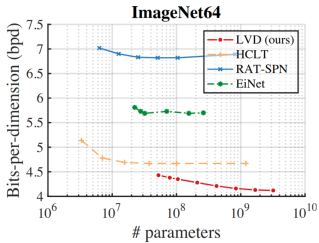
Learning recipe



1) Build a ***region graph***

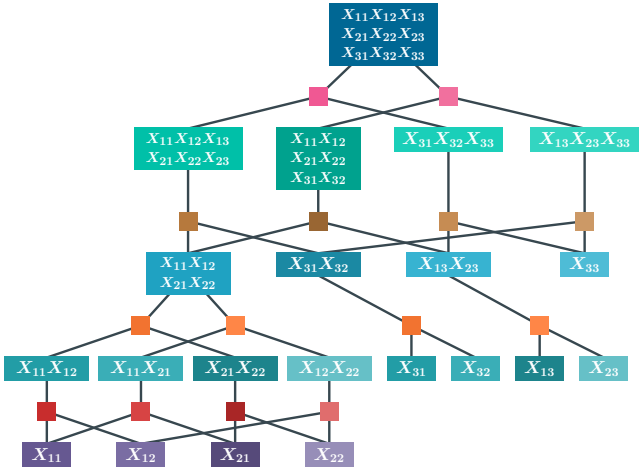


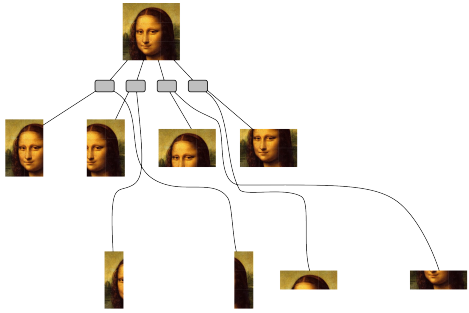
2) Overparameterize

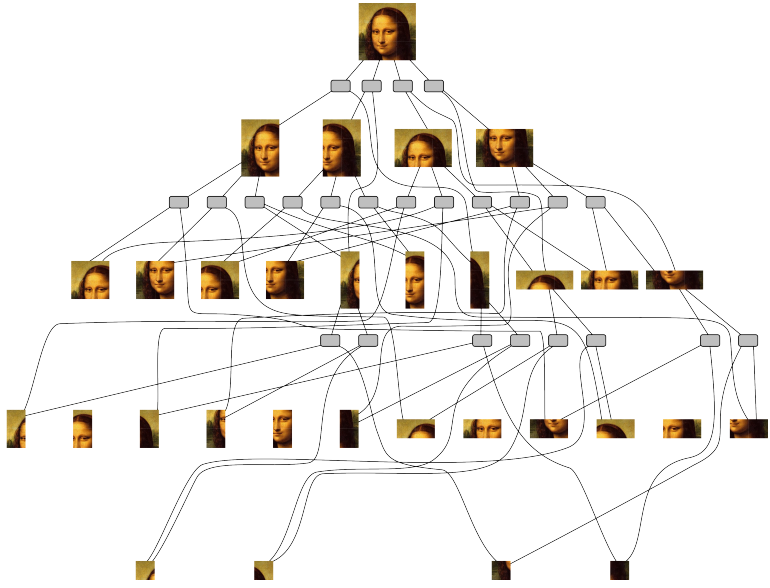


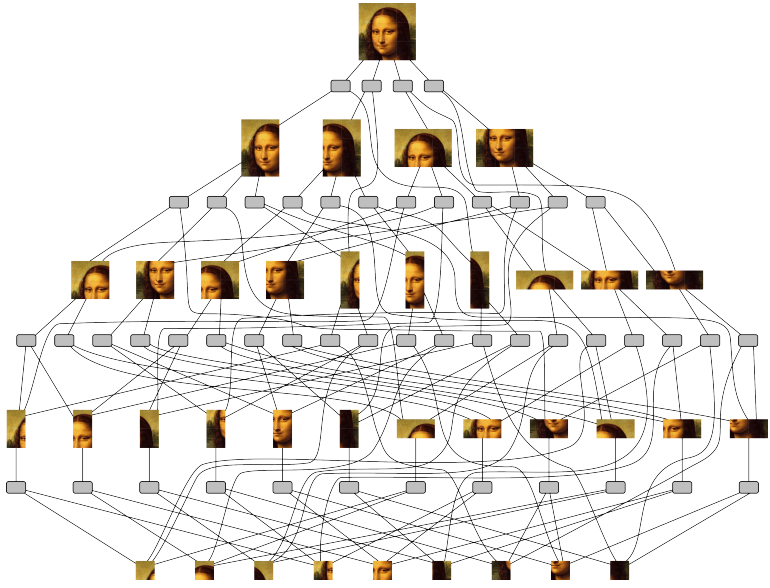
3) Learn parameters

Which region graph?





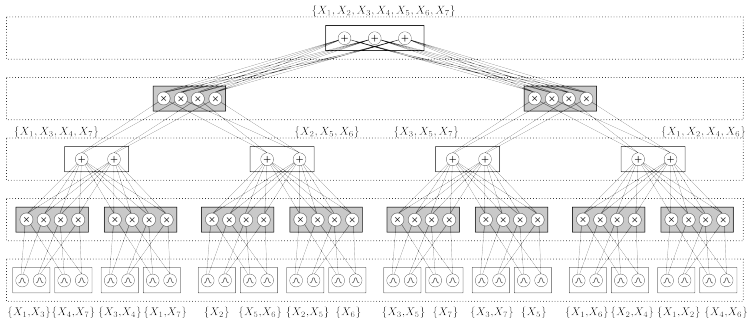


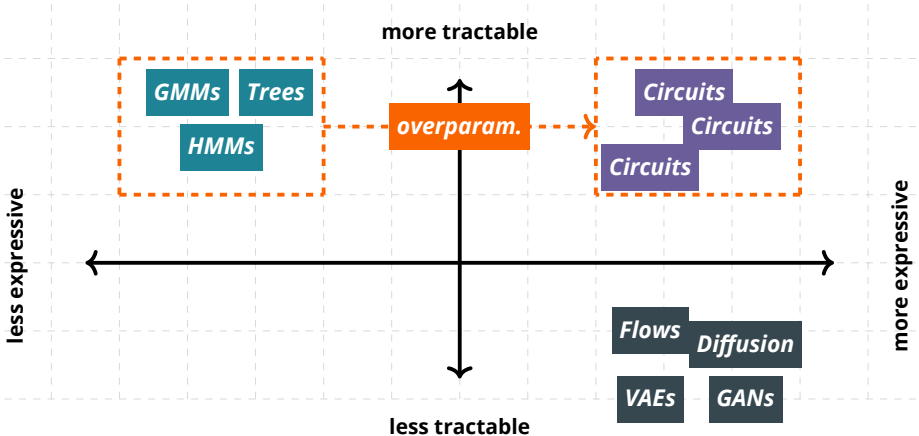


Random regions graphs

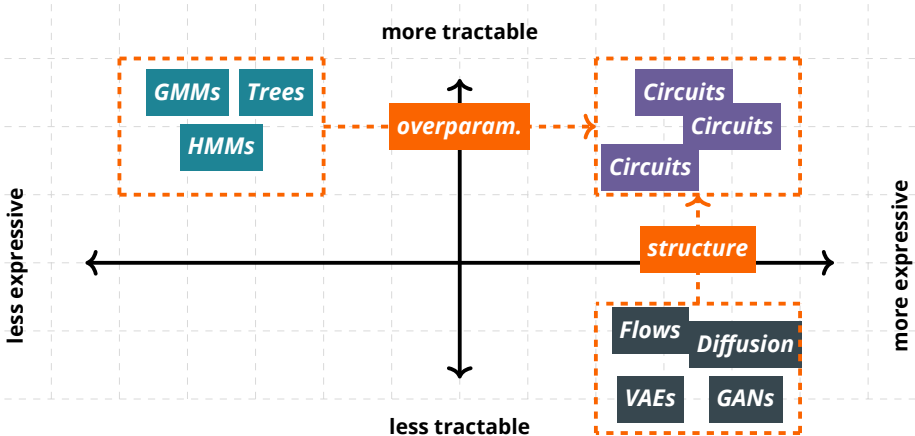
The "no-learning" option *[Peharz et al. 2019]*

Generating a random region graph, by recursively splitting $\mathbf{X}$ into two random parts:

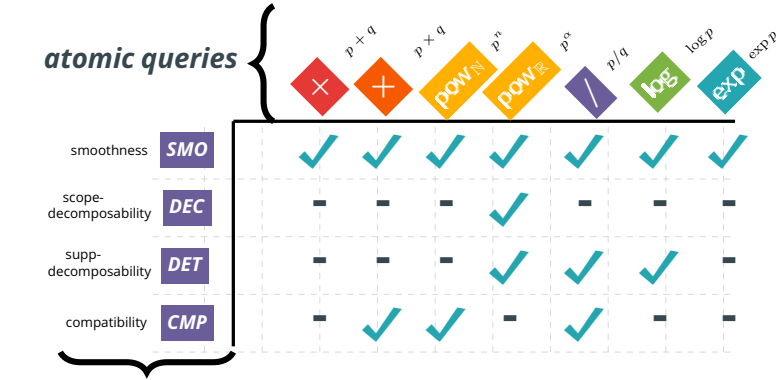




then make it more expressive!



impose structure!



what really matters is structure!