

probabilistic circuits

from inference to learning

antonio vergari (he/him)



@tetraduzione

robert peharz

16th July 2024 - ESSAI 2024 - Athens

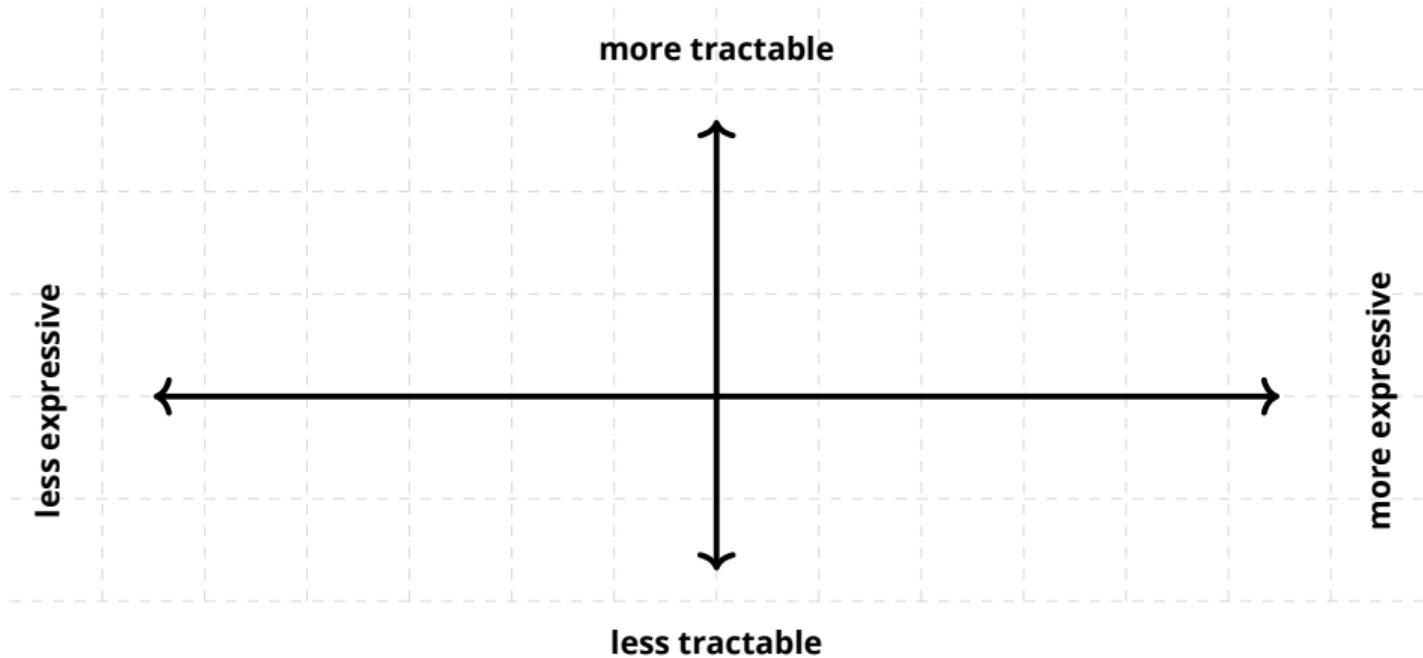
slides available at

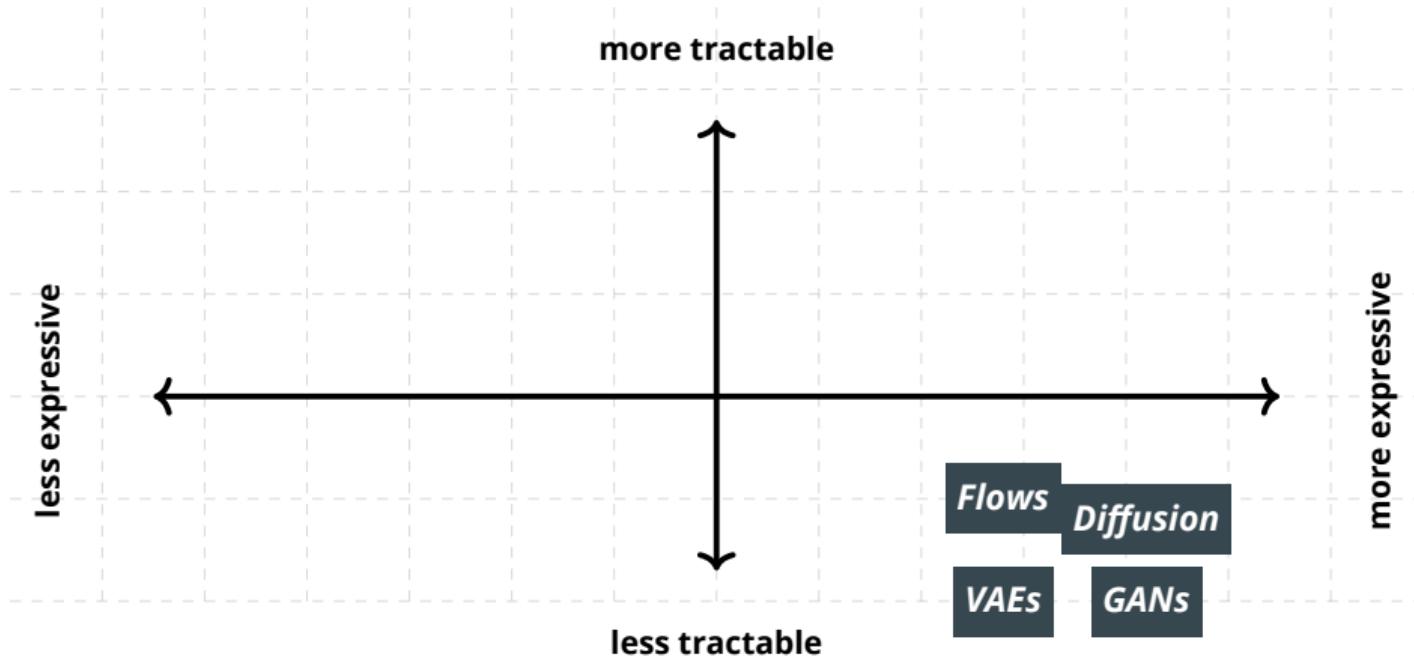


deep generative models

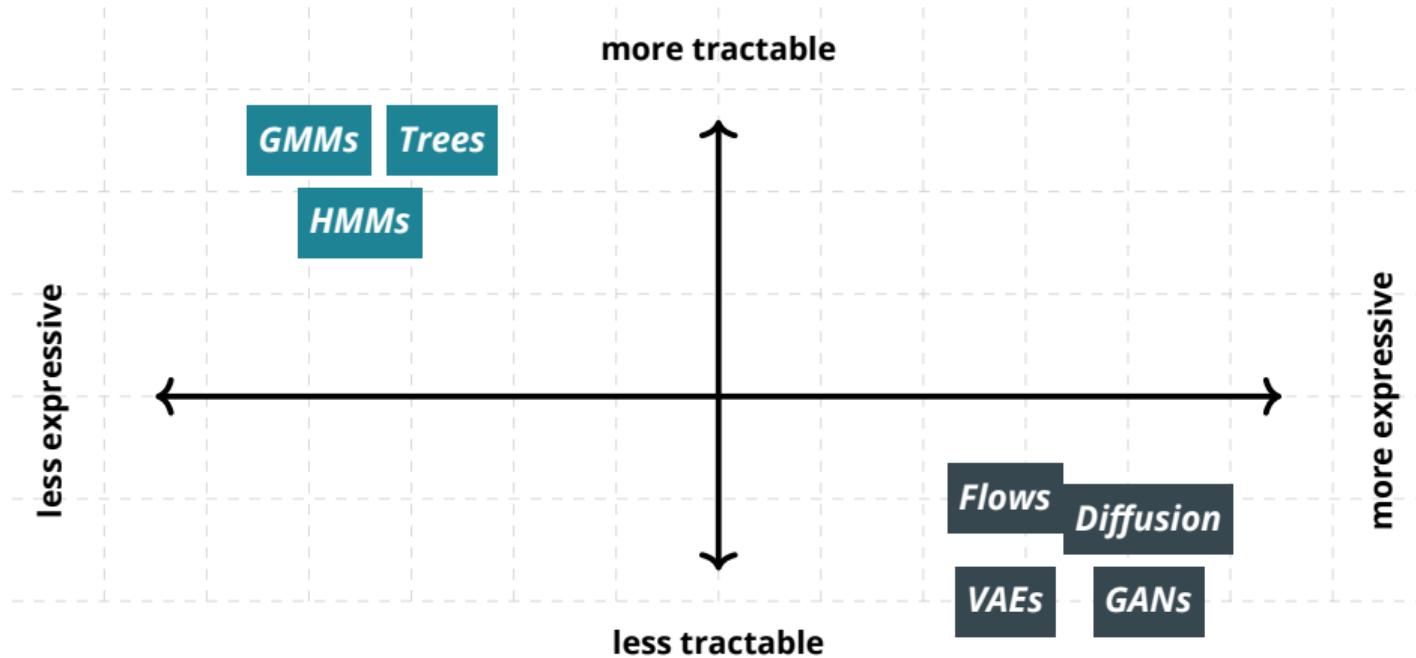
+

*flexible and reliable
logic & probabilistic reasoning?*

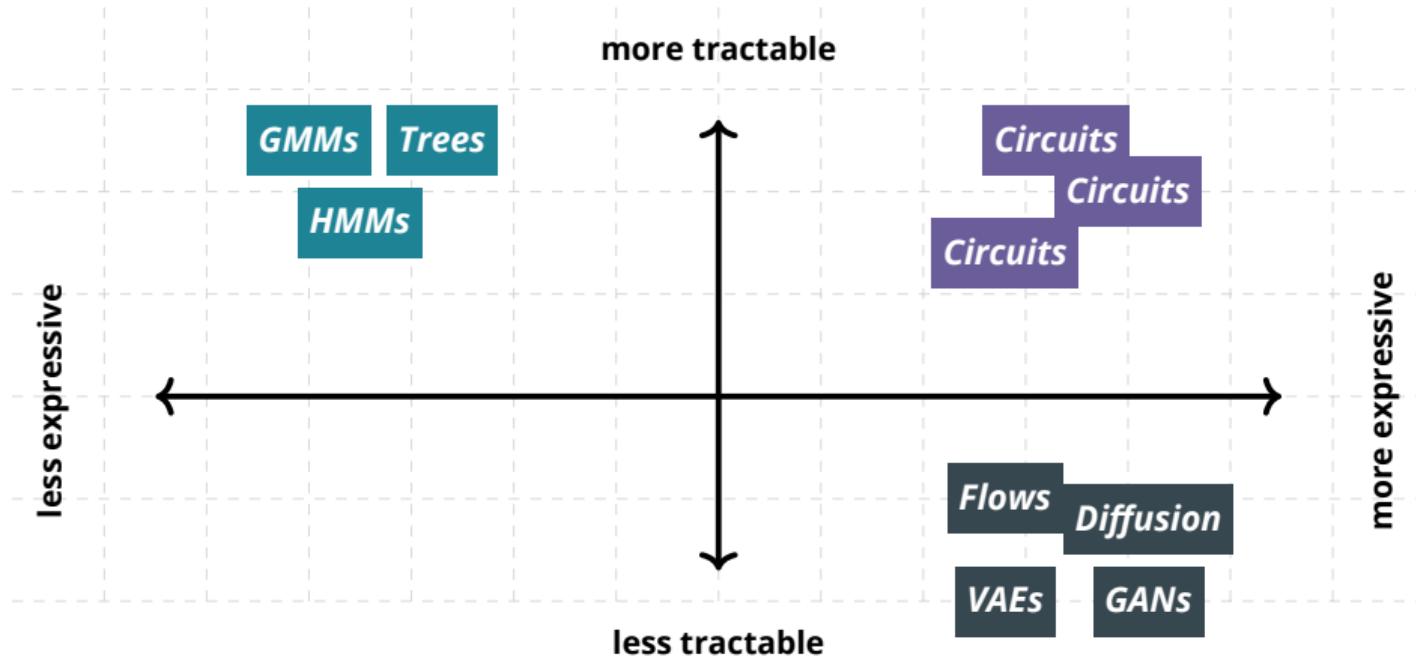




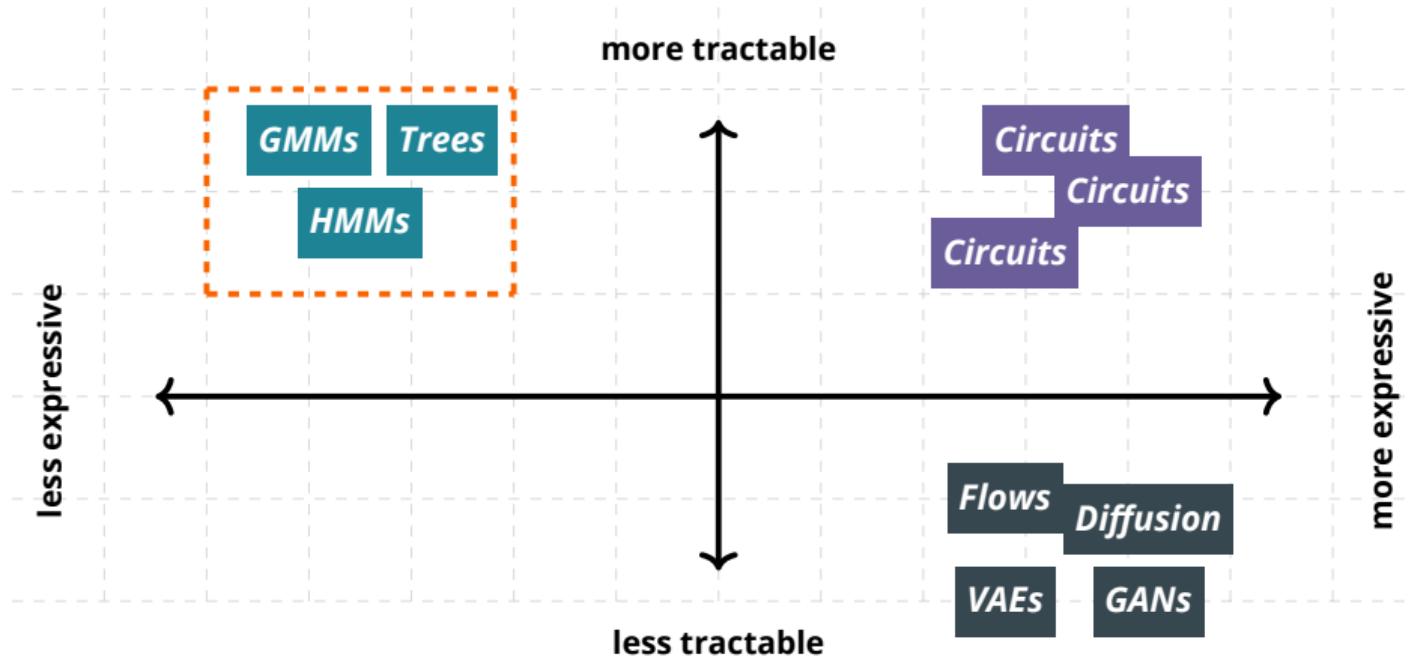
expressive models are not much tractable...



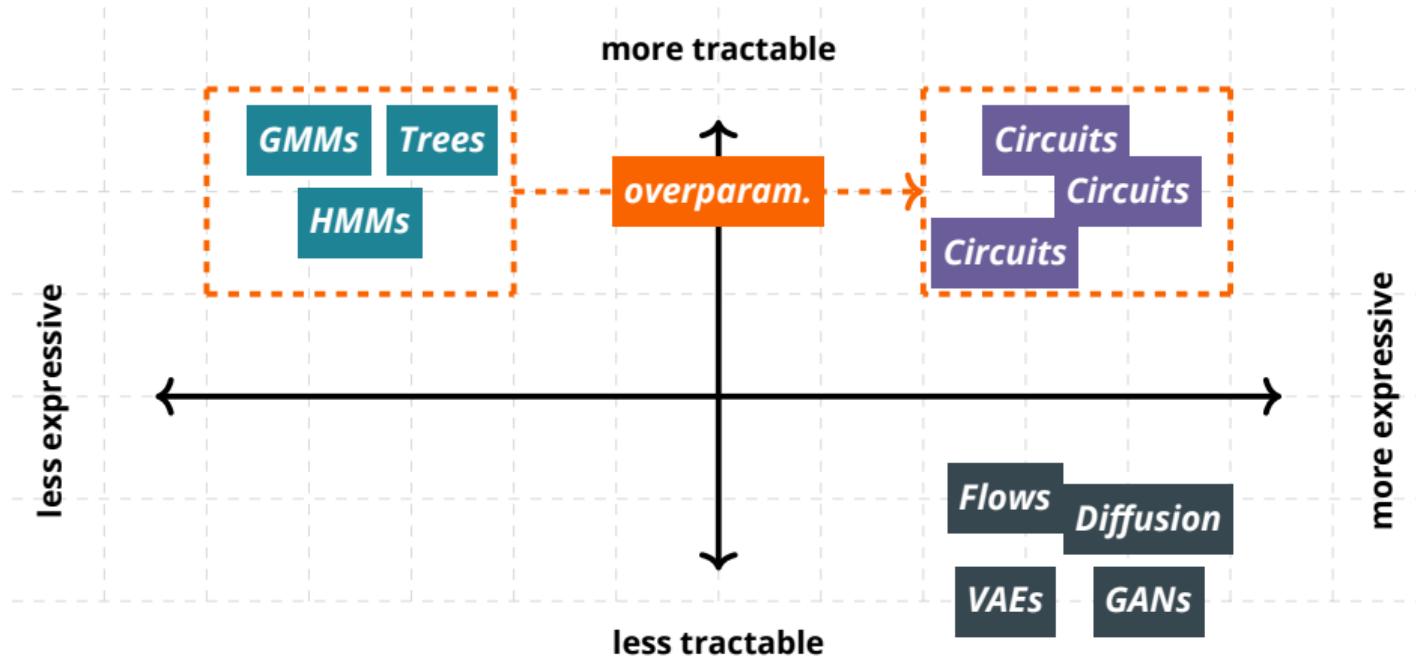
tractable models are not that expressive...



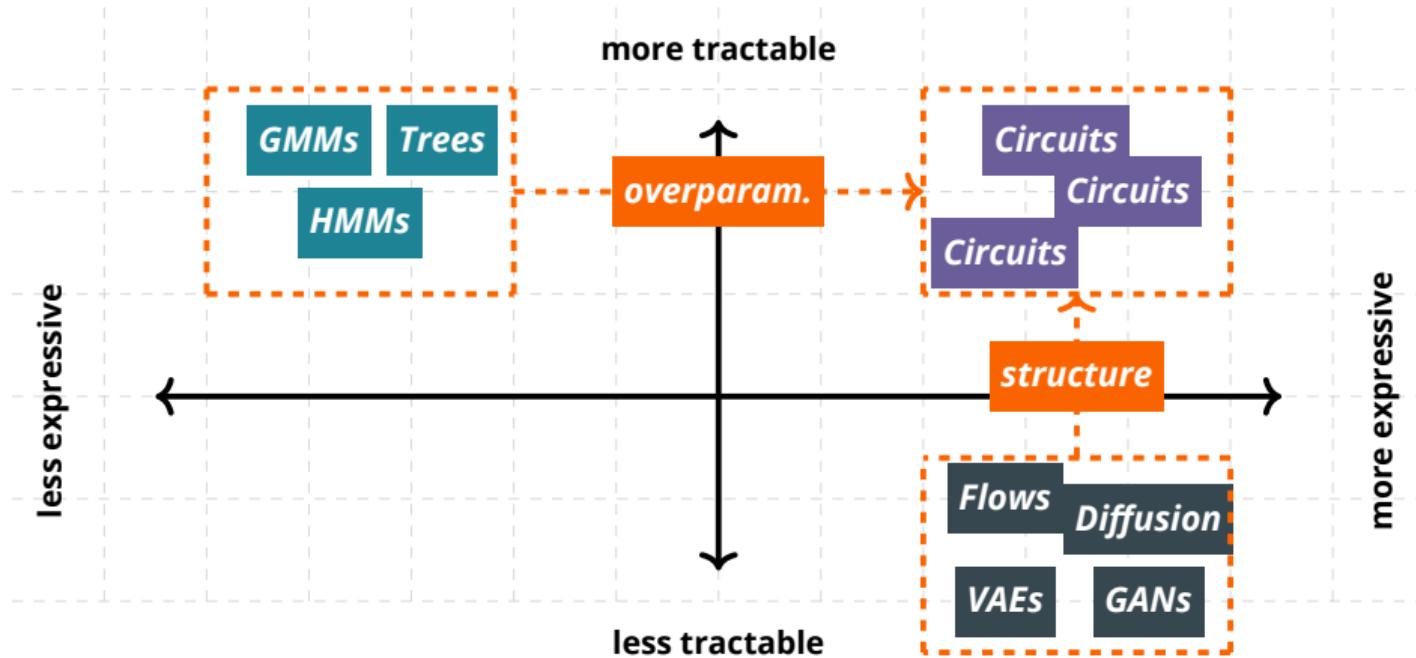
circuits can be both expressive and tractable!



start simple...



then make it more expressive!



impose structure!

Probabilistic Circuits (PCs)

A grammar for tractable computational graphs

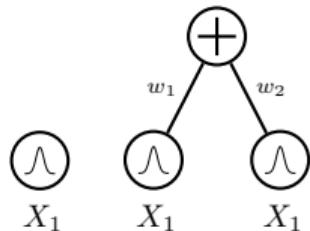
I. *A simple tractable function is a circuit*

$$\bigcirc \wedge \\ X_1$$

Probabilistic Circuits (PCs)

A grammar for tractable computational graphs

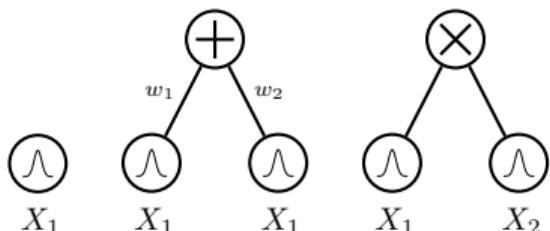
- I. A simple tractable function is a circuit
- II. A weighted combination of circuits is a circuit



Probabilistic Circuits (PCs)

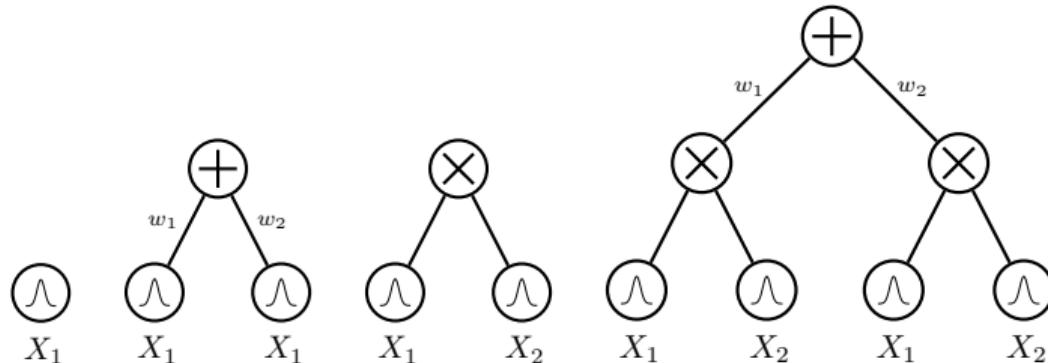
A grammar for tractable computational graphs

- 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



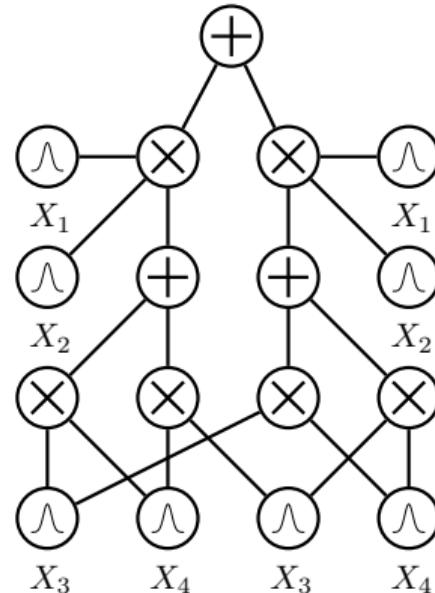
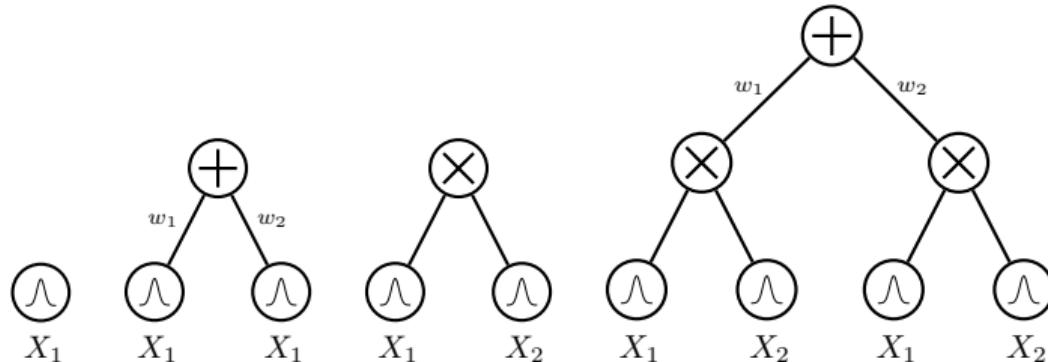
Probabilistic Circuits (PCs)

A grammar for tractable computational graphs



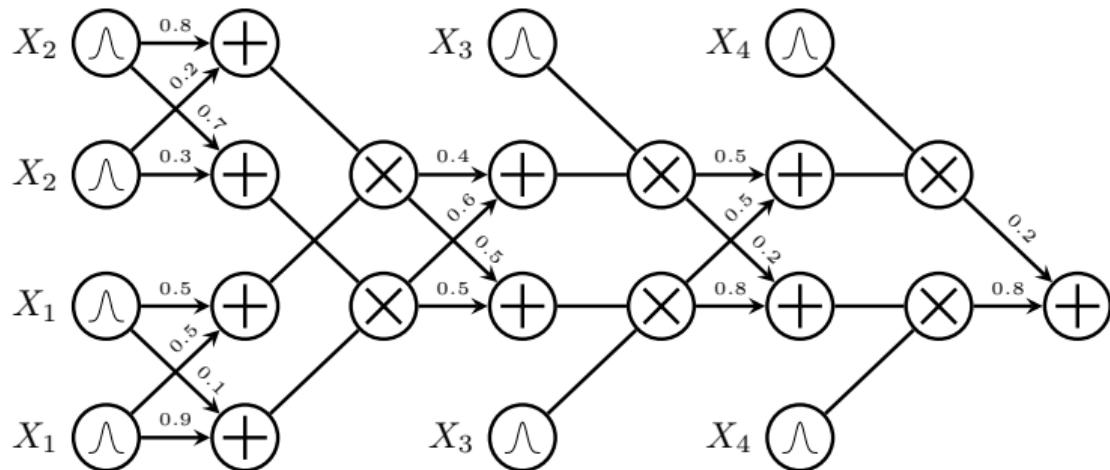
Probabilistic Circuits (PCs)

A grammar for tractable computational graphs



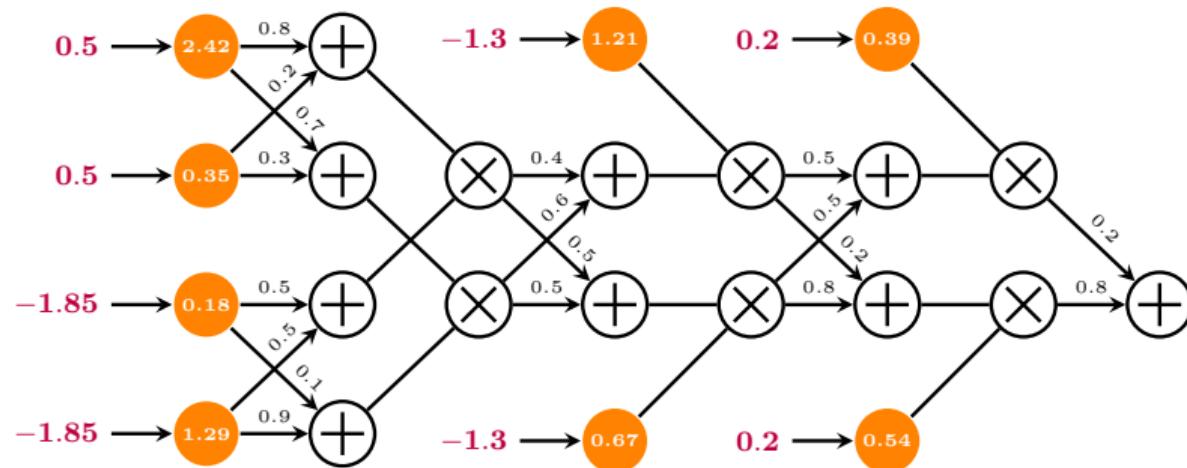
Probabilistic queries = feedforward evaluation

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



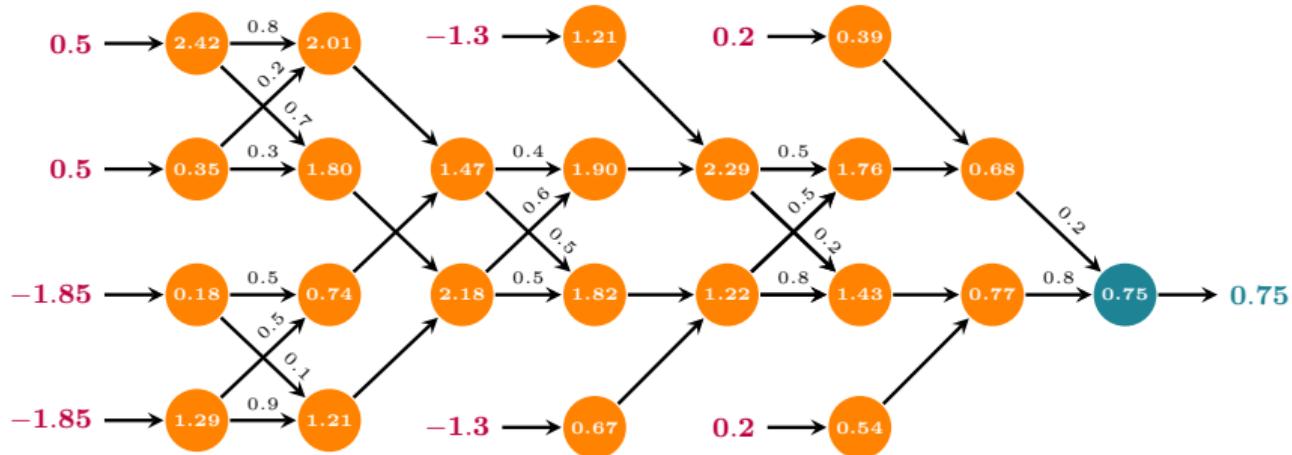
Probabilistic queries = feedforward evaluation

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



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?

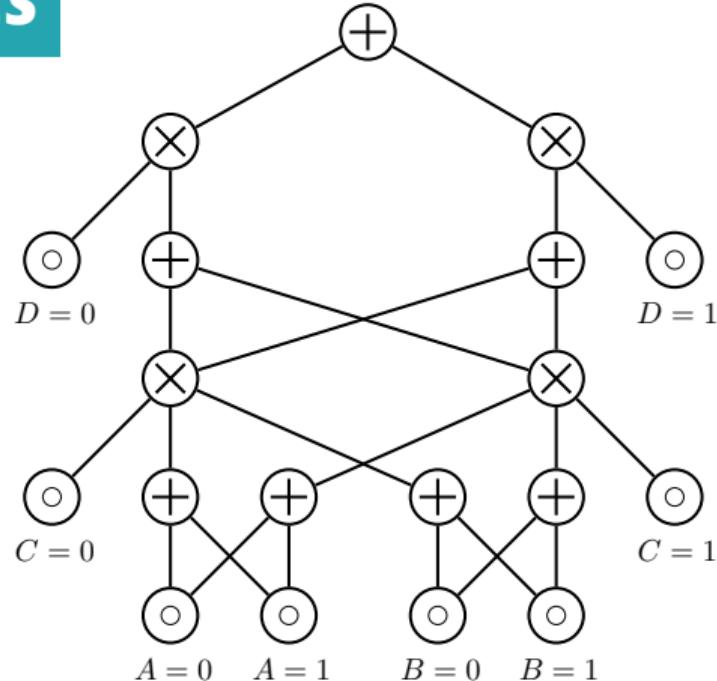
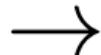
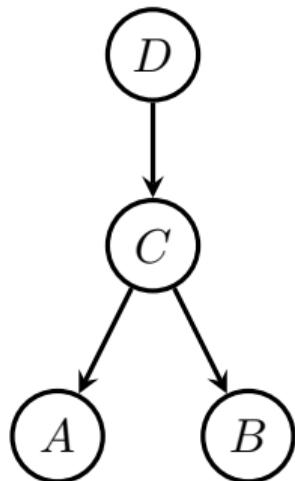
1. A grammar for tractable models

One formalism to represent many probabilistic and logical models

⇒ #HMMs #Trees #XGBoost, Tensor Networks, ...
and other PGMs...

from PGMs to circuits

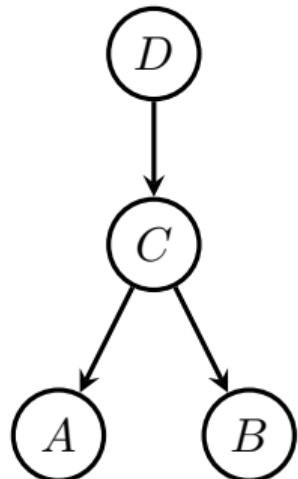
via compilation



from PGMs to circuits

via compilation

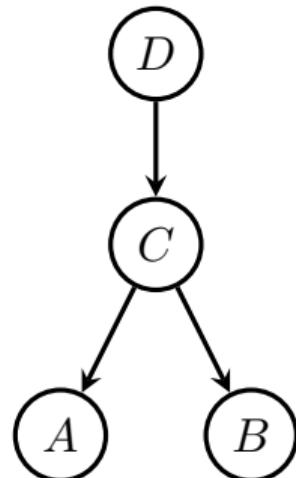
Bottom-up **compilation**: starting from leaves...



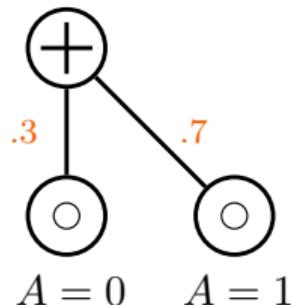
from PGMs to circuits

via compilation

...compile a leaf CPT



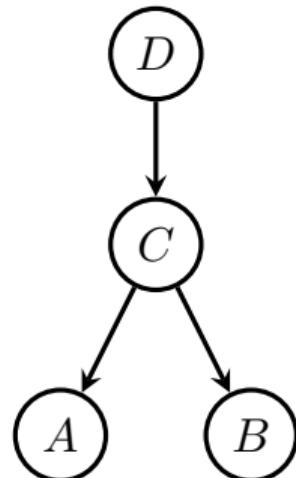
$$p(A|C = 0)$$



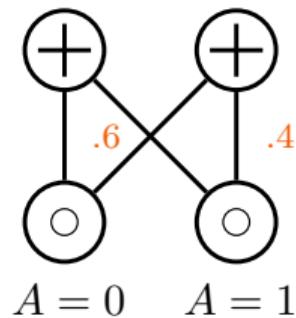
from PGMs to circuits

via compilation

...compile a leaf CPT



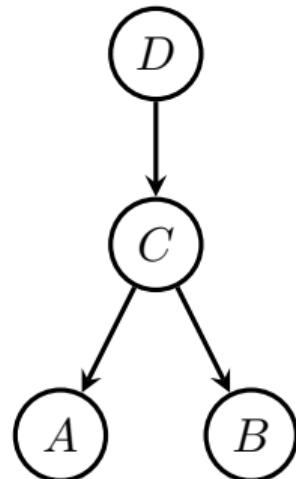
$$p(A|C = 1)$$



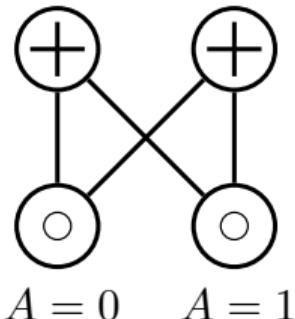
from PGMs to circuits

via compilation

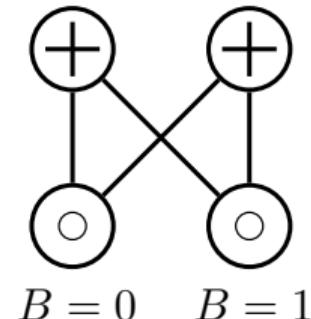
...compile a leaf CPT...for all leaves...



$$p(A|C)$$



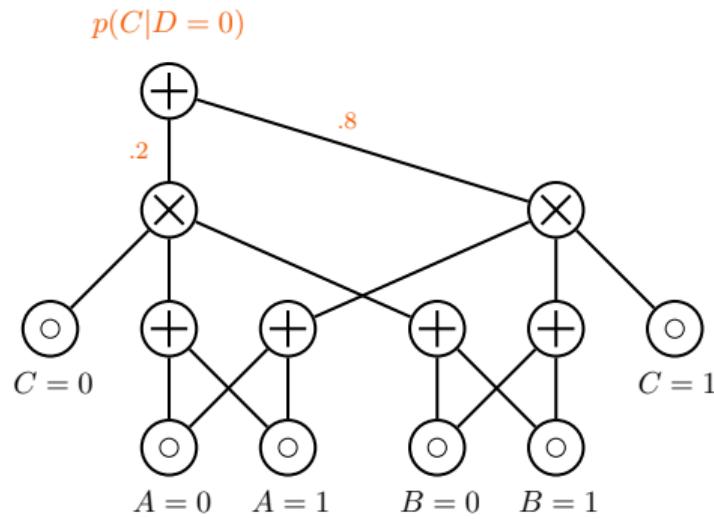
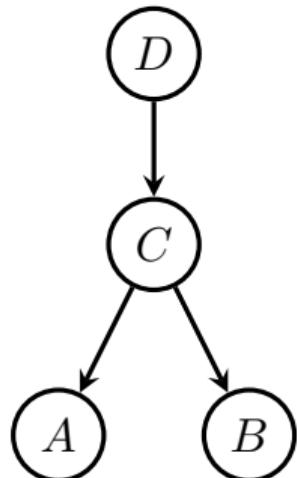
$$p(B|C)$$



from PGMs to circuits

via compilation

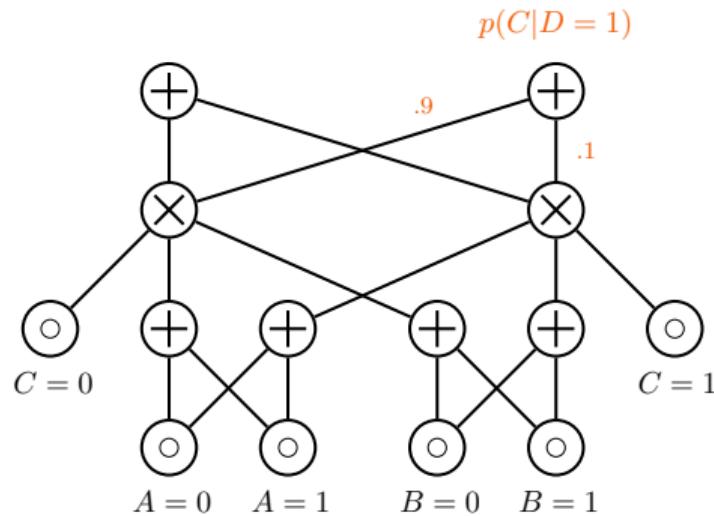
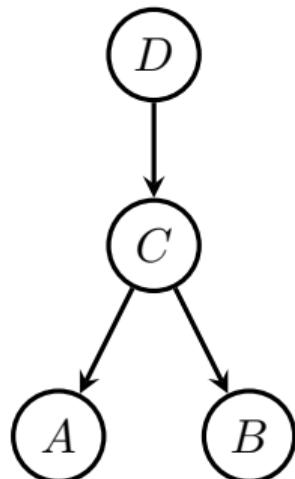
...and recurse over parents...



from PGMs to circuits

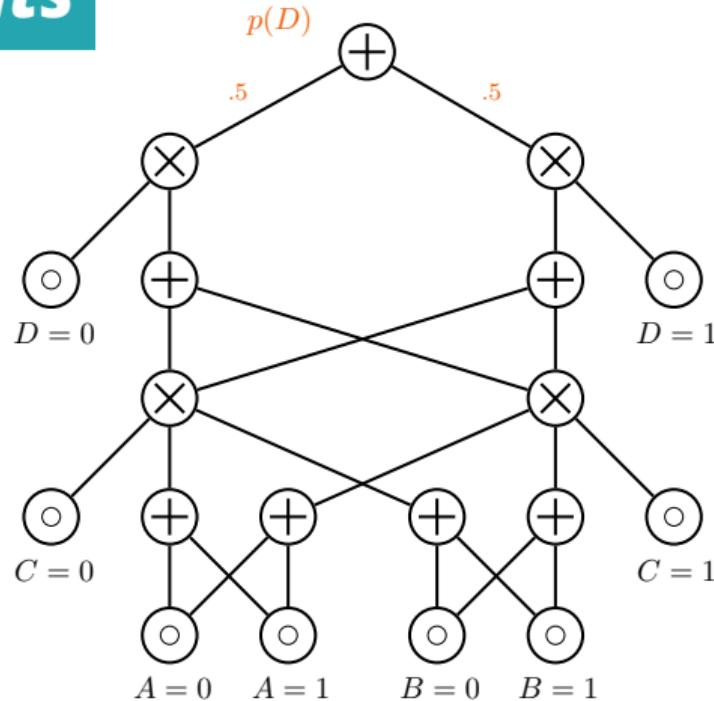
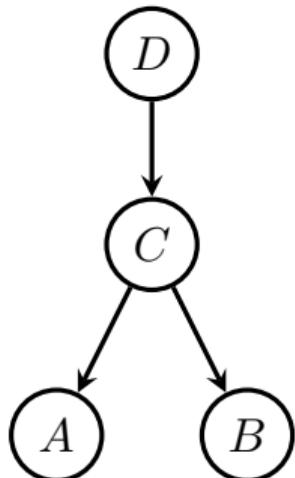
via compilation

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



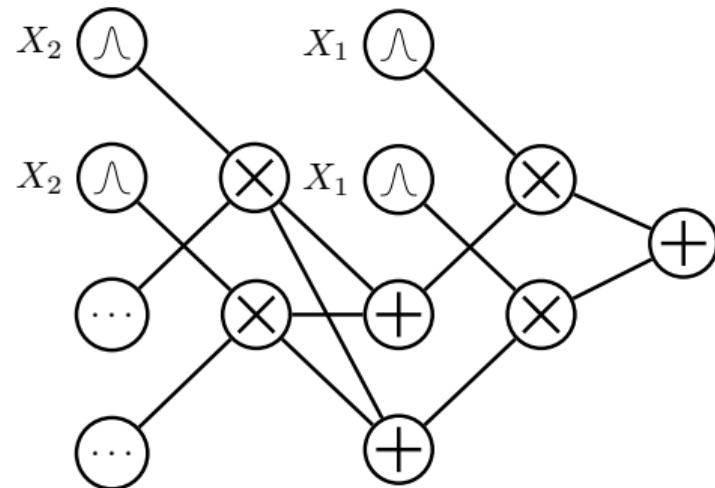
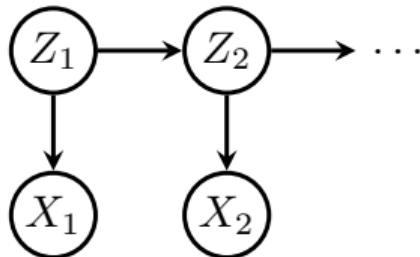
from PGMs to circuits

via compilation



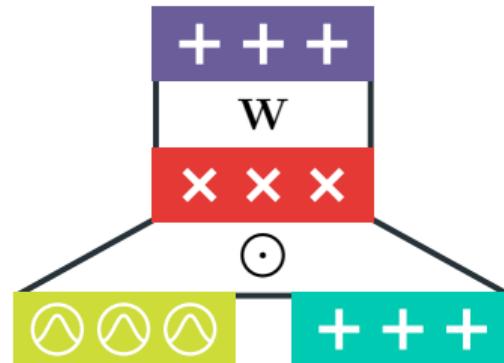
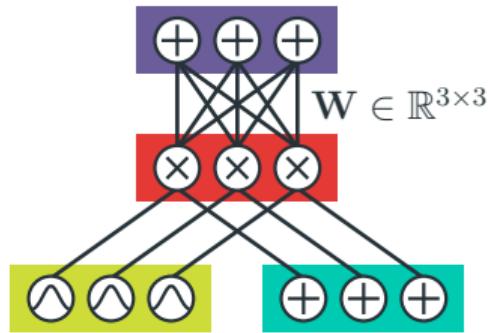
HMMs

as computational graphs



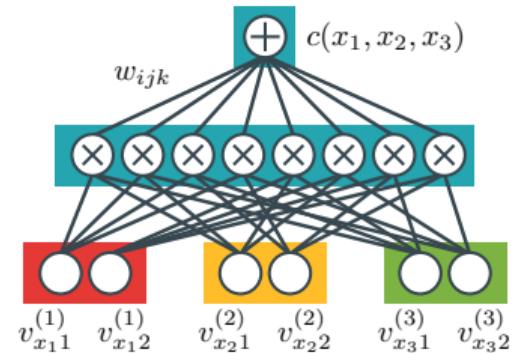
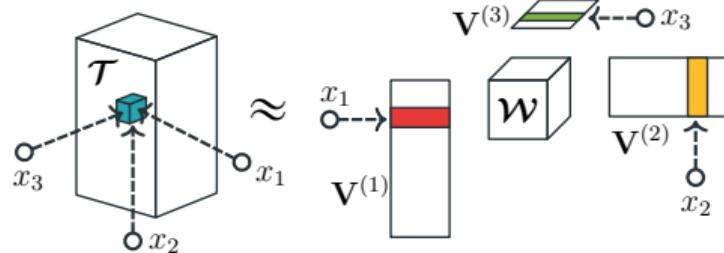
overparameterize

and tensorize



tensor factorizations

as circuits



...why PCs?

1. A grammar for tractable models

One formalism to represent many probabilistic and logical models

⇒ #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

⇒ #HMMs #Trees #XGBoost, Tensor Networks, ...
and other PGMs...

2. Expressiveness

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

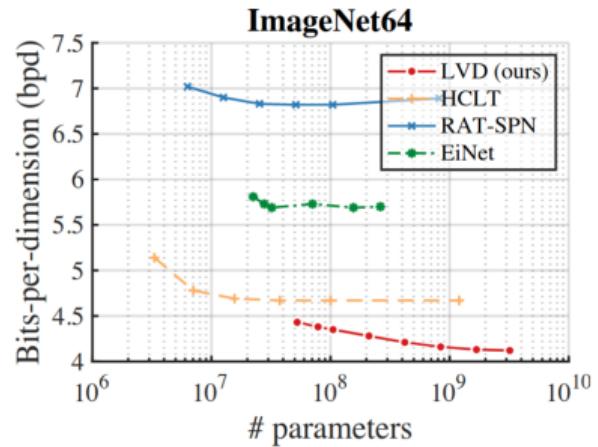
how expressive?

	QPQ	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!

how scalable?

Dataset	TPMs				DGMs		
	LVD (ours)	HCLT	EiNet	RAT-SPN	Glow	RealNVP	BIVA
ImageNet32	4.39\pm0.01	4.82	5.63	6.90	4.09	4.28	3.96
ImageNet64	4.12\pm0.00	4.67	5.69	6.82	3.81	3.98	-
CIFAR	4.38\pm0.02	4.61	5.81	6.95	3.35	3.49	3.08



up to billions of parameters

...why PCs?

1. A grammar for tractable models

One formalism to represent many probabilistic and logical models

⇒ #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

Structural properties

property A

property B

property C

property D

Structural properties

property A

tractable computation of *arbitrary integrals*

property B

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

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

property C

property D

\Rightarrow tractable partition function

Structural properties

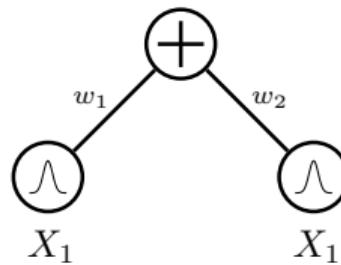
smoothness

the inputs of sum units are defined over the same variables

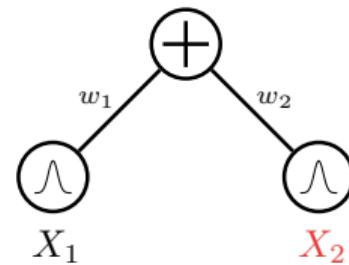
decomposability

compatibility

determinism



smooth circuit



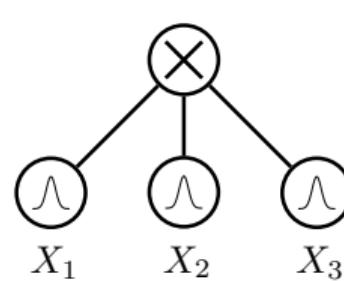
non-smooth circuit

Structural properties

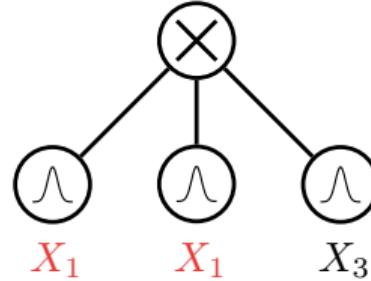
smoothness

the inputs of prod units are defined over disjoint variable sets

decomposability



compatibility

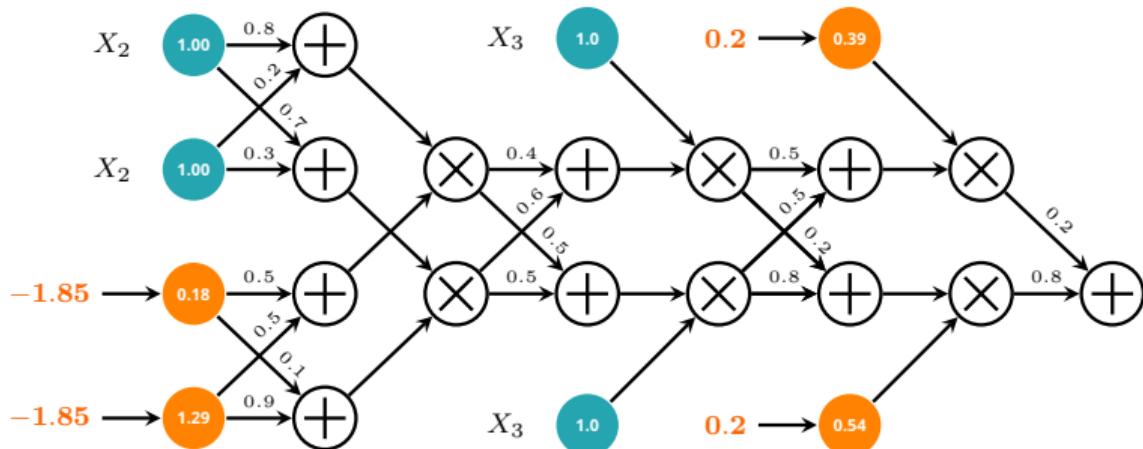


determinism

decomposable circuit non-decomposable circuit

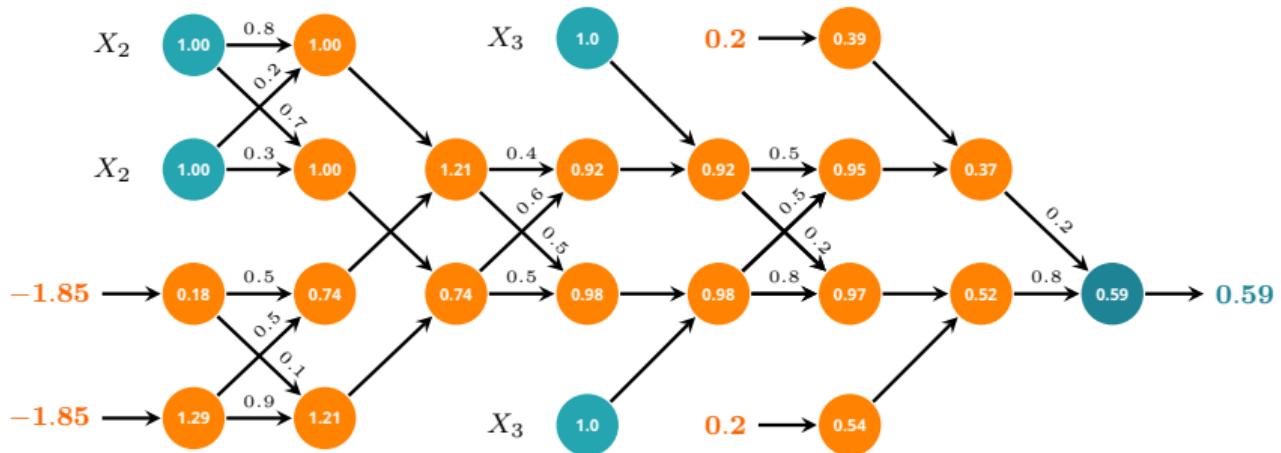
Probabilistic queries = feedforward evaluation

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



Probabilistic queries = feedforward evaluation

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



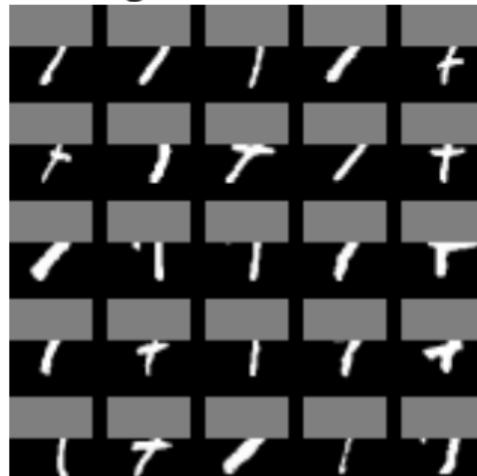
Tractable inference on PCs

Einsum networks

Original

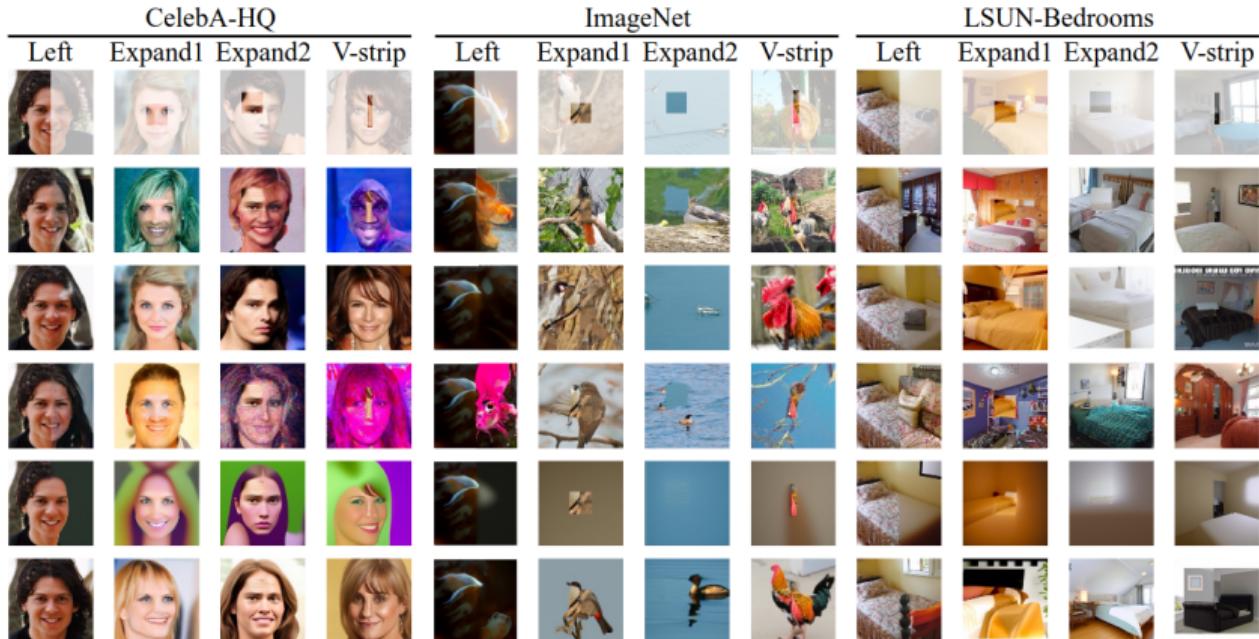


Missing



Conditional sample





general expectations

Integrals involving two or more functions:

$$\int \textcolor{orange}{p}(\mathbf{x}) \textcolor{teal}{f}(\mathbf{x}) d \mathbf{X}$$



general expectations

Integrals involving two or more functions:

$$\int \textcolor{orange}{p}(\mathbf{x}) \textcolor{teal}{f}(\mathbf{x}) d\mathbf{X}$$

represent both $\textcolor{orange}{p}$ and $\textcolor{teal}{f}$ as circuits...but with which structural properties?



general expectations

Integrals involving two or more functions:

$$\int \textcolor{orange}{p}(\mathbf{x}) \textcolor{teal}{f}(\mathbf{x}) d\mathbf{X}$$

represent both $\textcolor{orange}{p}$ and $\textcolor{teal}{f}$ as circuits...but with which structural properties?

E.g.,

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



Structural properties

smoothness

decomposability

compatibility

determinism

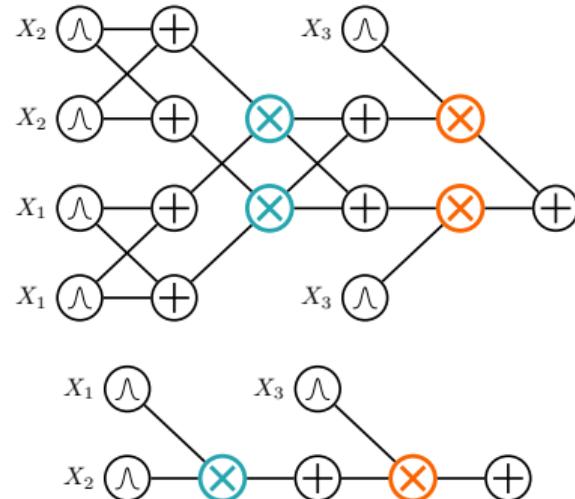
Structural properties

smoothness

decomposability

compatibility

determinism



compatible circuits

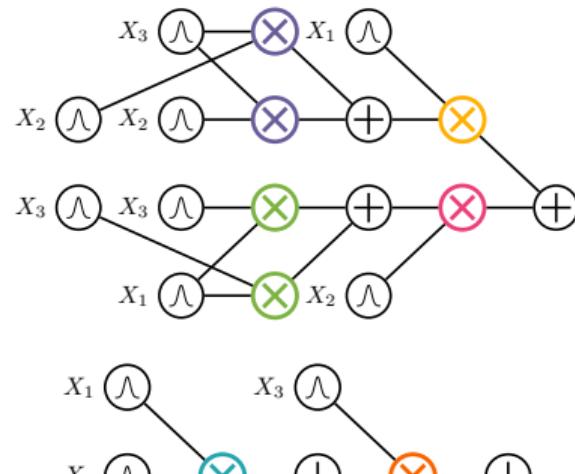
Structural properties

smoothness

decomposability

compatibility

determinism



non-compatible circuits

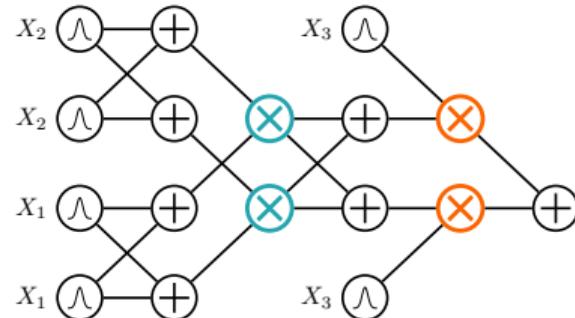
Structural properties

smoothness

decomposability

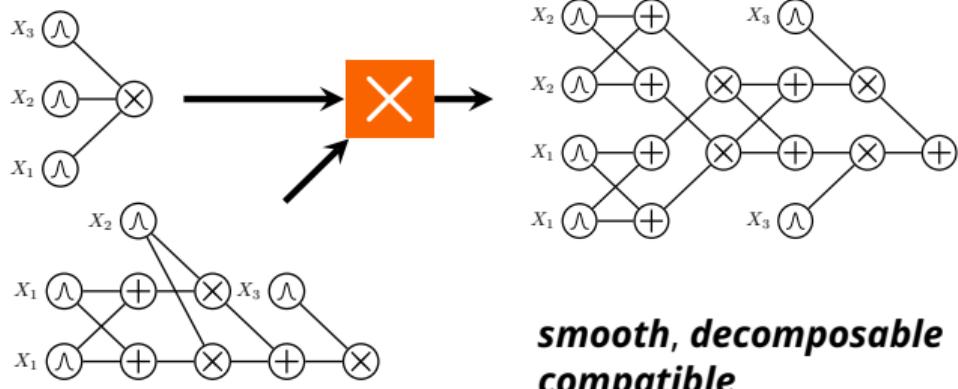
compatibility

determinism



a circuit that is compatible with itself is called
structured decomposable

Tractable products

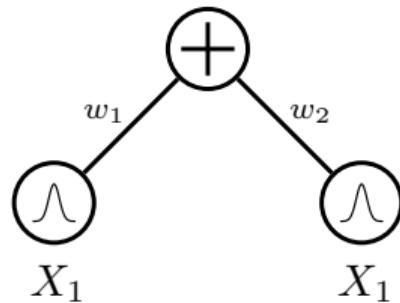


exactly compute $\int \mathbf{p}(\mathbf{x}) \mathbf{f}(\mathbf{x}) d\mathbf{X}$ **in time** $O(|\mathbf{p}| |\mathbf{f}|)$

Tractable products

an intuition why/how

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



Tractable products

an intuition why/how

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

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

Tractable products

an intuition why/how

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

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

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

cartesian product of “local” mixtures

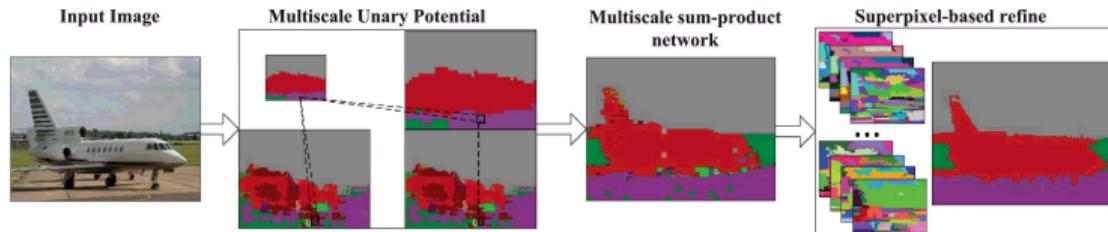
Maximum-a-posteriori (MAP) Inference

aka Most probable explanation (MPE)

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

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

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", 2016

Friesen et al., "Submodular Sum-product Networks for Scene Understanding", 2016

Structural properties

smoothness

decomposability

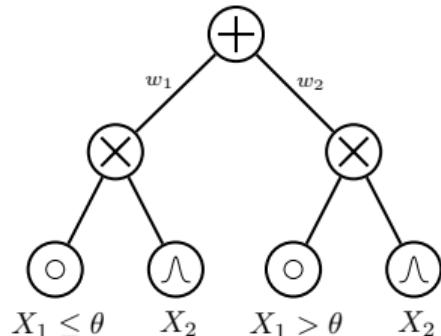
compatibility

determinism

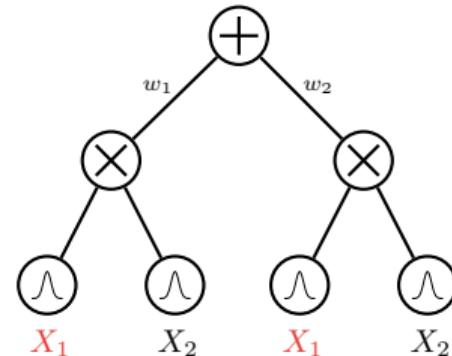
determinism

aka support-decomposability

A sum unit is deterministic if its inputs have disjoint supports



deterministic circuit



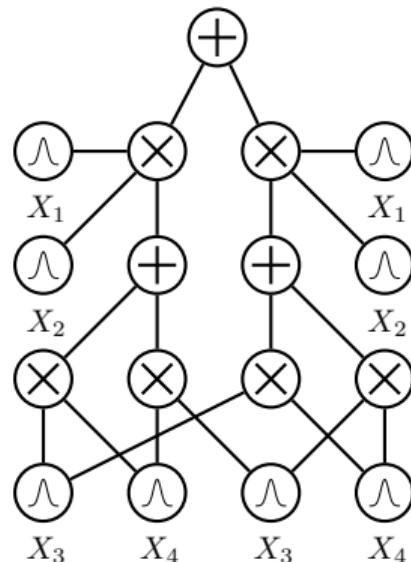
non-deterministic circuit

determinism + **decomposability** = **tractable MAP**

Computing maximization with arbitrary evidence e
⇒ *linear in circuit size!*

E.g., suppose we want to compute:

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

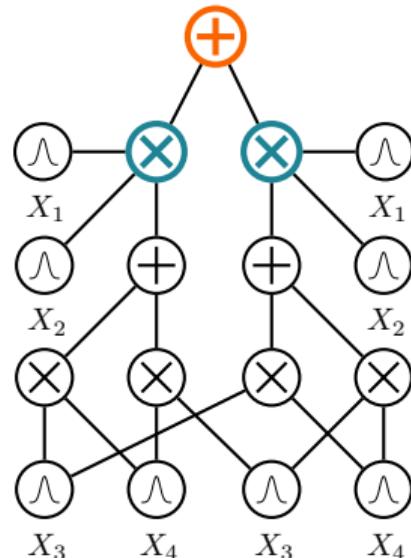


determinism + **decomposability** = **tractable MAP**

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

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

⇒ one non-zero term, thus sum is max

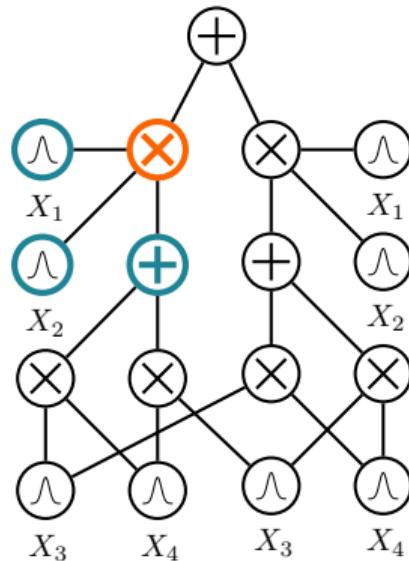


determinism + **decomposability** = **tractable MAP**

If $\mathbf{p}(\mathbf{q}, \mathbf{e}) = \mathbf{p}(\mathbf{q}_x, \mathbf{e}_x, \mathbf{q}_y, \mathbf{e}_y) = \mathbf{p}(\mathbf{q}_x, \mathbf{e}_x)\mathbf{p}(\mathbf{q}_y, \mathbf{e}_y)$
(decomposable product unit):

$$\begin{aligned}\max_{\mathbf{q}} \mathbf{p}(\mathbf{q} \mid \mathbf{e}) &= \max_{\mathbf{q}} \mathbf{p}(\mathbf{q}, \mathbf{e}) \\ &= \max_{\mathbf{q}_x, \mathbf{q}_y} \mathbf{p}(\mathbf{q}_x, \mathbf{e}_x, \mathbf{q}_y, \mathbf{e}_y) \\ &= \max_{\mathbf{q}_x} \mathbf{p}(\mathbf{q}_x, \mathbf{e}_x) \cdot \max_{\mathbf{q}_y} \mathbf{p}(\mathbf{q}_y, \mathbf{e}_y)\end{aligned}$$

⇒ solving optimization independently



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

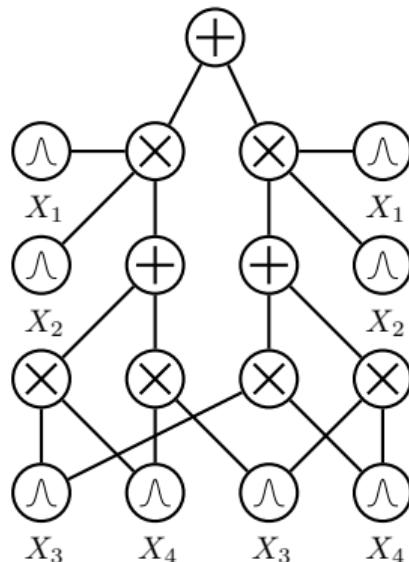
input distributions into max distributions

feedforward evaluation for

$\text{max}_{x_1, x_3} p(x_1, x_3 \mid x_2, x_4)$

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

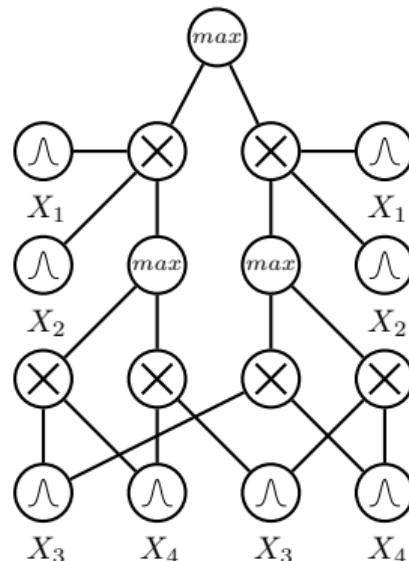
input distributions into max distributions

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

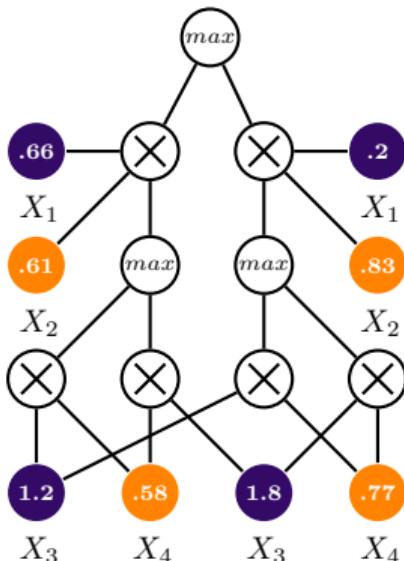
input distributions into max distributions

feedforward evaluation for

$\text{max}_{x_1, x_3} p(x_1, x_3 \mid x_2, x_4)$

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

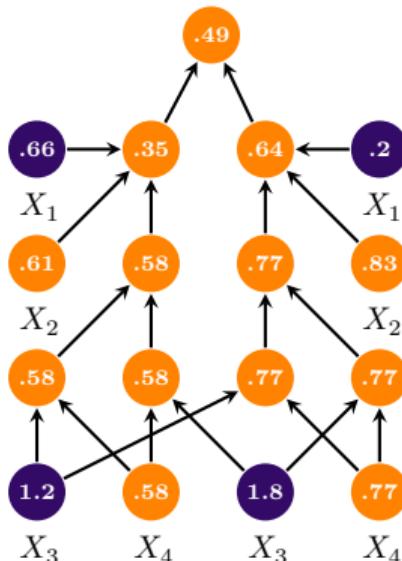
input distributions into max distributions

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

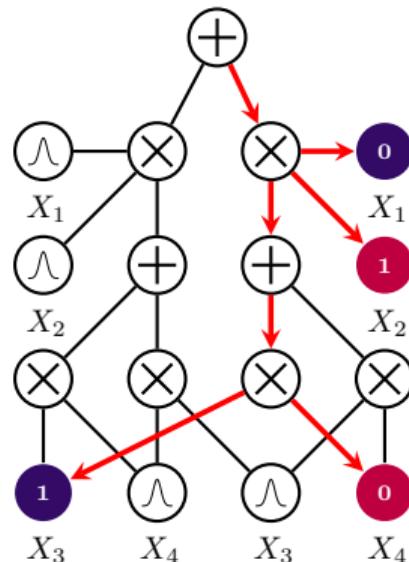
input distributions into max distributions

feedforward evaluation for

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

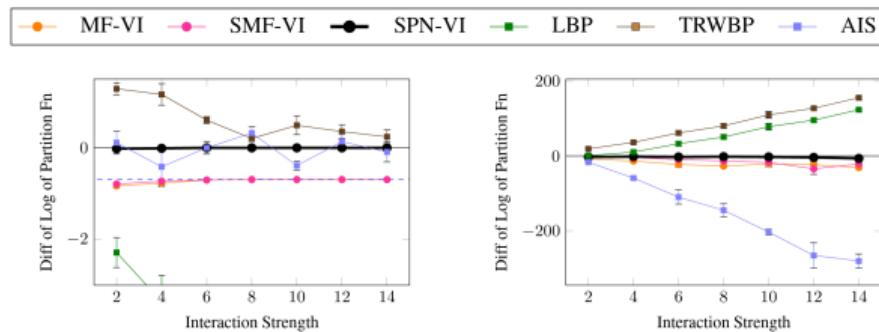
retrieve max activations in backward pass

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



Determinism + decomposability = tractable ELBO

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



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

Shih et al., "Probabilistic Circuits for Variational Inference in Discrete Graphical Models", *NeurIPS*, 2020

Goal

all this inference is cool...

but how can we **learn** these circuits?

learning probabilistic circuits

Probabilistic circuits are (peculiar) neural networks...

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?

How to enforce and exploit structural properties?

Learning probabilistic circuits

	<i>Parameters</i>	<i>Structure</i>
<i>Generative</i>	?	?
<i>Discriminative</i>	?	?

Maximum likelihood

the go-to objective in ProbML

Given a dataset $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ and your parametric model $p_\theta(\mathbf{X})$ solve

$$\hat{\theta}_{\text{ML}} = \max_{\theta} \prod_{i=1}^N p_\theta(\mathbf{x}^{(i)}) = \min_{\theta} - \sum_{i=1}^N \log p_\theta(\mathbf{x}^{(i)})$$

\Rightarrow minimize the negative log-likelihood (NLL)

which parameters?

how to reparameterize circuits

Input distributions.

Sum unit parameters.

which parameters?

how to reparameterize circuits

Input distributions. Each input can be a different parametric distribution

⇒ *Bernoullis, Categoricals, Gaussians, exponential families, small NNs, ...*

Sum unit parameters.

which parameters?

how to reparameterize circuits

Input distributions. Each input can be a different parametric distribution

Sum unit parameters. Enforce them to be non-negative, i.e., $w_i \geq 0$ but unnormalized

$$w_i = \exp(\alpha_i), \quad \alpha_i \in \mathbb{R}, \quad i = 1, \dots, K$$

and renormalize the loss

$$\min_{\theta} - \left(\sum_{i=1}^N \log \tilde{p}_{\theta}(\mathbf{x}^{(i)}) - \log \int \tilde{p}_{\theta}(\mathbf{x}^{(i)}) d\mathbf{X} \right)$$

or just renormalize the weights, i.e., $\sum_i w_i = 1$

$$\mathbf{w} = \text{softmax}(\boldsymbol{\alpha}), \quad \boldsymbol{\alpha} \in \mathbb{R}^K$$

learning input distributions

As simple as tossing a coin

$$\begin{array}{c} \textcircled{\text{A}} \\ X_1 \end{array}$$

The simplest PC: a single input distribution p_L with parameters θ

\Rightarrow maximum likelihood (ML) estimation over data \mathcal{D}

learning input distributions

As simple as tossing a coin

$$\begin{array}{c} \textcircled{\text{A}} \\ X_1 \end{array}$$

The simplest PC: a single input distribution p_L with parameters θ

\Rightarrow maximum likelihood (ML) estimation over data \mathcal{D}

E.g. Bernoulli with parameter θ

$$\hat{\theta}_{\text{ML}} = \frac{\sum_{x \in \mathcal{D}} \mathbf{1}[x = 1] + \alpha}{|\mathcal{D}| + 2\alpha} \quad \Rightarrow \text{Laplace smoothing}$$

learning input distributions

General case: still simple

Bernoulli, Gaussian, Dirichlet, Poisson, Gamma are **exponential families** of the form:

$$p_L(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \exp(\mathbf{T}(\mathbf{x})^T \boldsymbol{\theta} - A(\boldsymbol{\theta}))$$

learning input distributions

General case: still simple

Bernoulli, Gaussian, Dirichlet, Poisson, Gamma are **exponential families** of the form:

$$p_L(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \exp(\mathbf{T}(\mathbf{x})^T \boldsymbol{\theta} - A(\boldsymbol{\theta}))$$

Where:

$A(\boldsymbol{\theta})$: log-normalizer

$\mathbf{h}(\mathbf{x})$ base-measure

$\mathbf{T}(\mathbf{x})$ sufficient statistics

$\boldsymbol{\theta}$ natural parameters

learning input distributions

General case: still simple

Bernoulli, Gaussian, Dirichlet, Poisson, Gamma are **exponential families** of the form:

$$p_L(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \exp(\mathbf{T}(\mathbf{x})^T \boldsymbol{\theta} - A(\boldsymbol{\theta}))$$

Where:

$A(\boldsymbol{\theta})$: log-normalizer

$\mathbf{h}(\mathbf{x})$ base-measure

$\mathbf{T}(\mathbf{x})$ sufficient statistics

$\boldsymbol{\theta}$ natural parameters

or $\boldsymbol{\phi}$ expectation parameters — 1:1 mapping with $\boldsymbol{\theta} \Rightarrow \boldsymbol{\theta} = \boldsymbol{\theta}(\boldsymbol{\phi})$

learning input distributions

General case: still simple

Bernoulli, Gaussian, Dirichlet, Poisson, Gamma are **exponential families** of the form:

$$p_L(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \exp(\mathbf{T}(\mathbf{x})^T \boldsymbol{\theta} - A(\boldsymbol{\theta}))$$

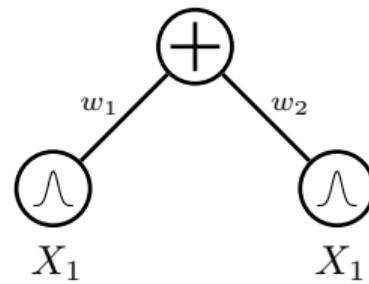
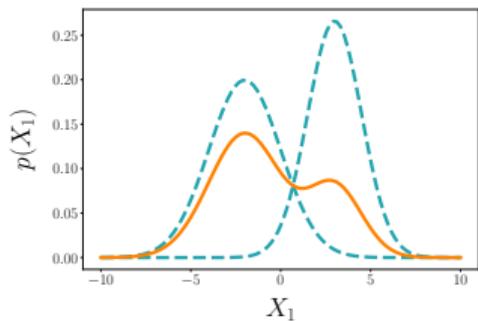
Maximum likelihood estimation is still “**counting**”:

$$\hat{\boldsymbol{\phi}}_{ML} = \mathbb{E}_{\mathcal{D}}[\mathbf{T}(\mathbf{x})] = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \mathbf{T}(\mathbf{x})$$

$$\hat{\boldsymbol{\theta}}_{ML} = \boldsymbol{\theta}(\hat{\boldsymbol{\phi}}_{ML})$$

a single sum unit

the simplest “real” PC

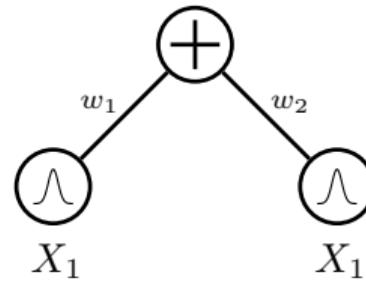
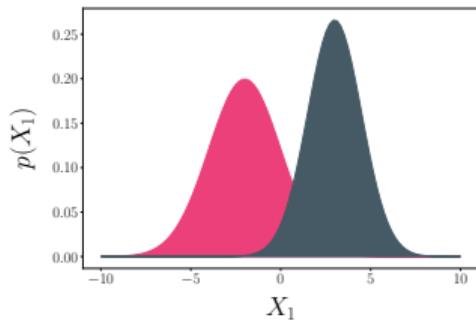


Recall that sum units represent ***mixture models***:

$$p_S(\mathbf{x}) = \sum_{k=1}^K w_k p_{L_k}(\mathbf{x})$$

a single sum unit

the simplest “real” PC

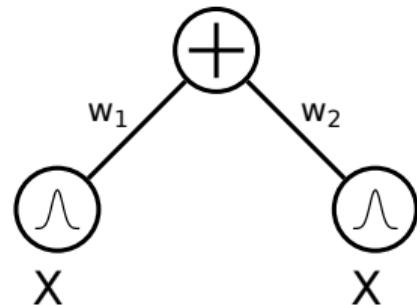


Recall that sum units represent ***latent variable models***:

$$p_S(\mathbf{x}) = \sum_{k=1}^K p(Z = k)p(\mathbf{x} \mid Z = k)$$

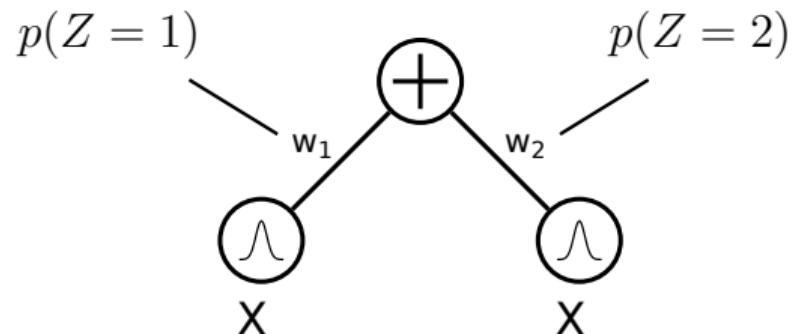
Expectation-Maximization

for deep mixtures/PCs



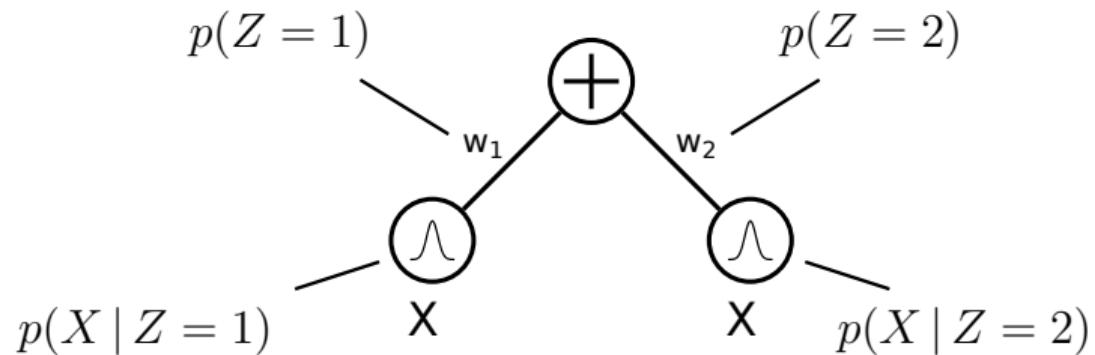
Expectation-Maximization

for deep mixtures/PCs



Expectation-Maximization

for deep mixtures/PCs



Expectation-Maximization

for shallow mixtures/PCs

ML if Z was observed:

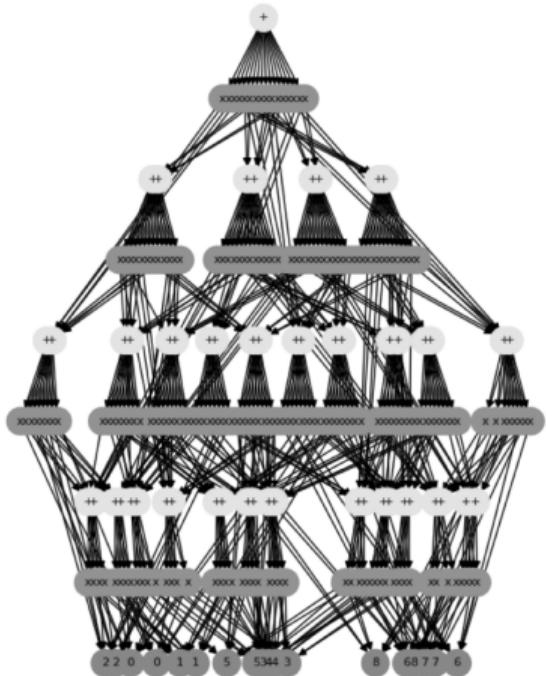
$$\hat{w}_k = \frac{\sum_{z \in \mathcal{D}} \mathbb{1}[z = k]}{|\mathcal{D}|} \quad \hat{\phi}_k = \frac{\sum_{\mathbf{x}, z \in \mathcal{D}} \mathbb{1}[z = k] T(\mathbf{x})}{\sum_{z \in \mathcal{D}} \mathbb{1}[z = k]}$$

Z is unobserved—but we have $p(Z = k \mid \mathbf{x}) \propto w_k \mathsf{L}_k(\mathbf{x})$.

$$w_k^{new} = \frac{\sum_{\mathbf{x} \in \mathcal{D}} p(Z = k \mid \mathbf{x})}{|\mathcal{D}|} \quad \phi_k^{new} = \frac{\sum_{\mathbf{x}, z \in \mathcal{D}} p(Z = k \mid \mathbf{x}) T(\mathbf{x})}{\sum_{z \in \mathcal{D}} p(Z = k \mid \mathbf{x})}$$

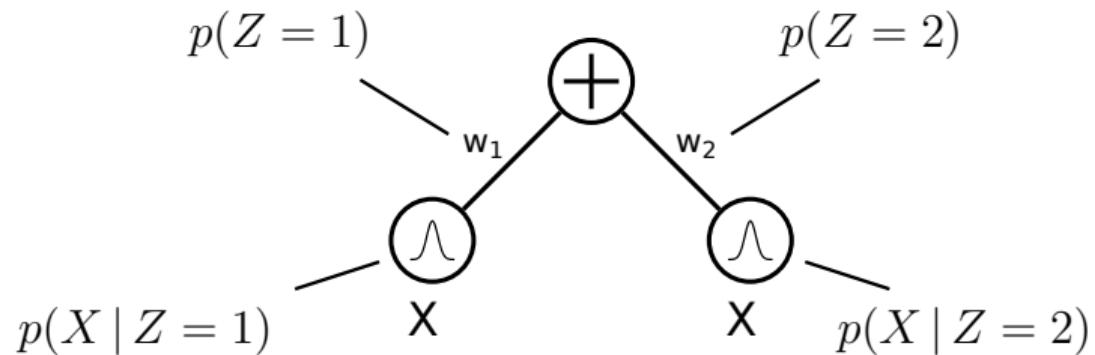
how to generalize...

EM to *deep* mixtures/circuits?



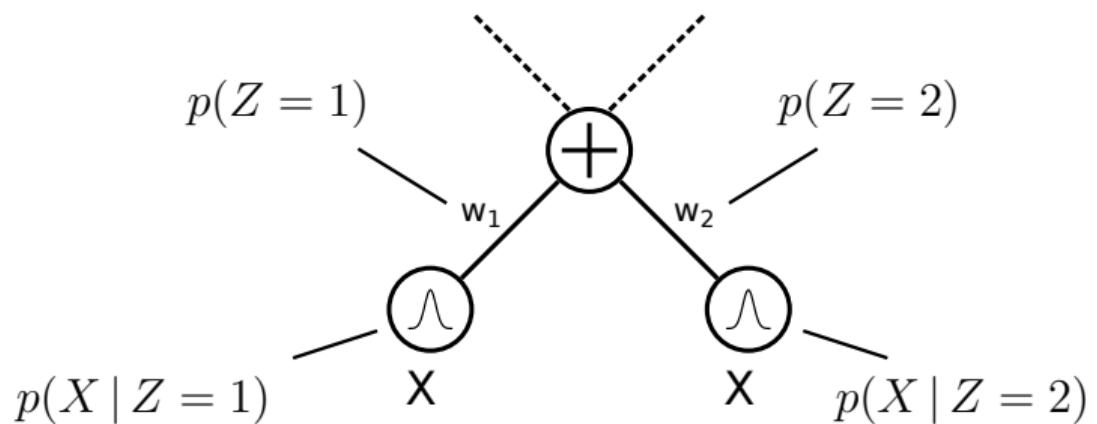
Expectation-Maximization

for deep mixtures/PCs



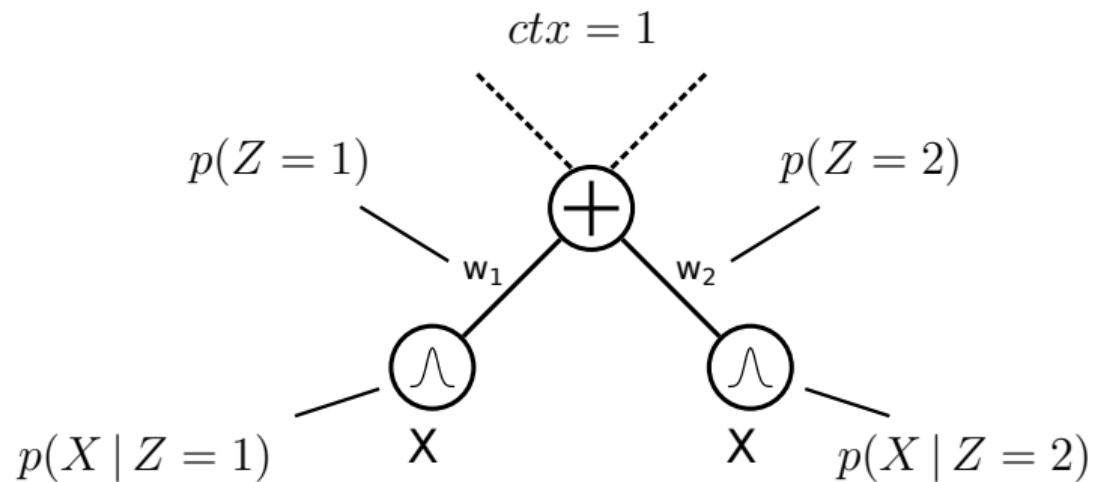
Expectation-Maximization

for deep mixtures/PCs



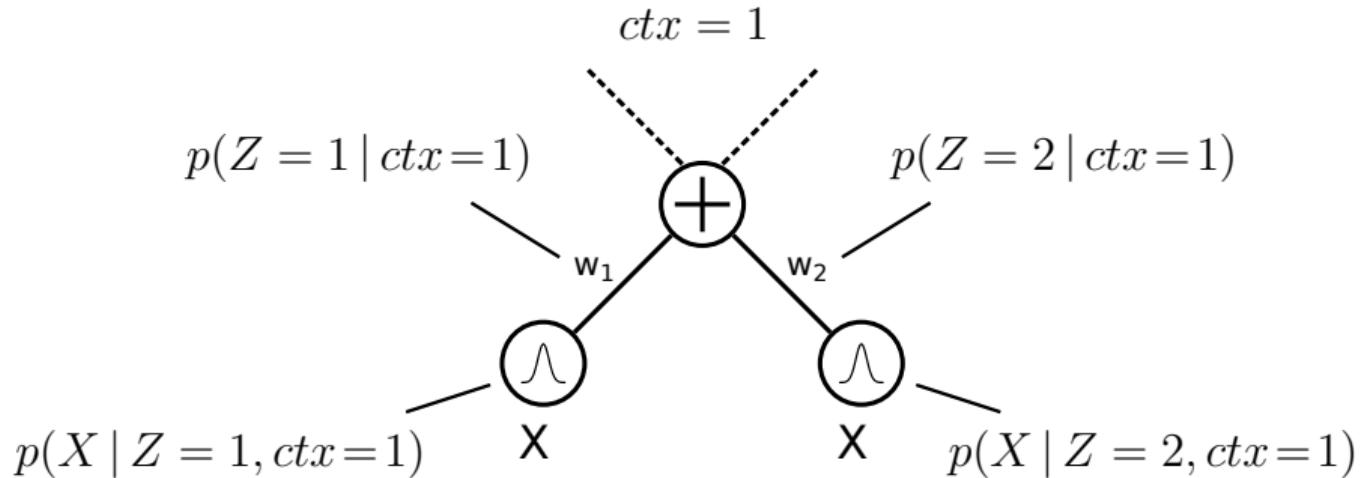
Expectation-Maximization

for deep mixtures/PCs



Expectation-Maximization

for deep mixtures/PCs



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:

$$p[ctx_i = 1, Z_i = j \mid \mathbf{x}; \mathbf{w}^{old}] = \frac{1}{p(\mathbf{x})} \frac{\partial p(\mathbf{x})}{\partial S_i(\mathbf{x})} N_j(\mathbf{x}) w_{i,j}^{old}$$

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 et al. 2016]

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

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

Learning probabilistic circuits

	<i>Parameters</i>	<i>Structure</i>
Generative	deterministic closed-form MLE [Kisa et al. 2014; Peharz et al. 2014]	
	non-deterministic EM [Poon et al. 2011; Peharz 2015; Zhao et al. 2016b] SGD [Sharir et al. 2016; Peharz et al. 2019b] Bayesian [Jaini et al. 2016; Rashwan et al. 2016] [Zhao et al. 2016a; Trapp et al. 2019; Vergari et al. 2019]	?
Discriminative	?	?

Learning probabilistic circuits

	<i>Parameters</i>	<i>Structure</i>
<i>Generative</i>	deterministic closed-form MLE [Kisa et al. 2014; Peharz et al. 2014]	greedy top-down [Gens et al. 2013; Rahman et al. 2014; Rooshenas et al. 2014] [Vergari et al. 2015] bottom-up [Peharz et al. 2013]
	non-deterministic EM [Poon et al. 2011; Peharz 2015; Zhao et al. 2016b] SGD [Sharir et al. 2016; Peharz et al. 2019b] Bayesian [Jaini et al. 2016; Rashwan et al. 2016] [Zhao et al. 2016a; Trapp et al. 2019; Vergari et al. 2019]	hill climbing [Lowd et al. 2008, 2013; Peharz et al. 2014] [Dennis et al. 2015; Liang et al. 2017; Galindez Olascoaga et al. 2019] [Dang et al. 2022] random RAT-SPNs [Peharz et al. 2019b] XCNet [Di Mauro et al. 2017]
<i>Discriminative</i>	?	?

Learning probabilistic circuits

	<i>Parameters</i>	<i>Structure</i>
<i>Generative</i>	<i>deterministic</i> closed-form MLE [Kisa et al. 2014; Peharz et al. 2014]	<i>greedy</i> top-down [Gens et al. 2013; Rahman et al. 2014; Rooshenas et al. 2014] [Vergari et al. 2019]
	<i>non-deterministic</i> EM [Poon et al. 2011; Peharz 2015; Zhao et al. 2016b] SGD [Sharir et al. 2016; Peharz et al. 2019b] Bayesian [Jaini et al. 2016; Rashwan et al. 2016] [Zhao et al. 2016a; Trapp et al. 2019; Vergari et al. 2019]	<i>hill climb</i> [Dennis et al. 1967] [Dang et al. 2022] a single pipeline [Dang et al. 2019] <i>random</i> RAT-SPNs [Peharz et al. 2019b] XCNet [Di Mauro et al. 2017]
<i>Discriminative</i>	?	?

Probabilistic Circuits (PCs)

the unit-wise definition

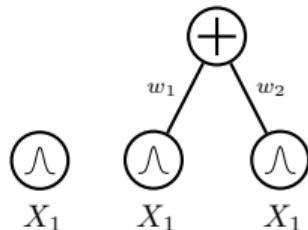
I. A simple tractable function is a circuit

$$\bigcirc \wedge \\ X_1$$

Probabilistic Circuits (PCs)

the unit-wise definition

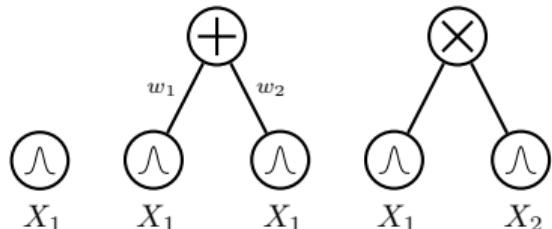
- I. A simple tractable function is a circuit
- II. A weighted combination of circuits is a circuit



Probabilistic Circuits (PCs)

the unit-wise definition

- 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)

the layer-wise definition

- I. A set of tractable functions is a circuit layer



Probabilistic Circuits (PCs)

the layer-wise definition

- I. A set of tractable functions is a circuit layer
- II. A linear projection of a layer is a circuit layer

$$c(\mathbf{x}) = \mathbf{W}l(\mathbf{x})$$



Probabilistic Circuits (PCs)

the layer-wise definition

- I. A set of tractable functions is a circuit layer
- II. A linear projection of a layer is a circuit layer

$$c(\mathbf{x}) = \mathbf{W}\mathbf{l}(\mathbf{x})$$



Probabilistic Circuits (PCs)

the layer-wise definition

- I. A set of tractable functions is a circuit layer
- II. A linear projection of a layer is a circuit layer
- III. The product of two layers is a circuit layer

$$c(\mathbf{x}) = \mathbf{l}(\mathbf{x}) \odot \mathbf{r}(\mathbf{x}) \quad // \text{ Hadamard}$$



Probabilistic Circuits (PCs)

the layer-wise definition

- I. A set of tractable functions is a circuit layer
- II. A linear projection of a layer is a circuit layer
- III. The product of two layers is a circuit layer

$$c(\mathbf{x}) = \mathbf{l}(\mathbf{x}) \odot \mathbf{r}(\mathbf{x}) \quad // \text{ Hadamard}$$

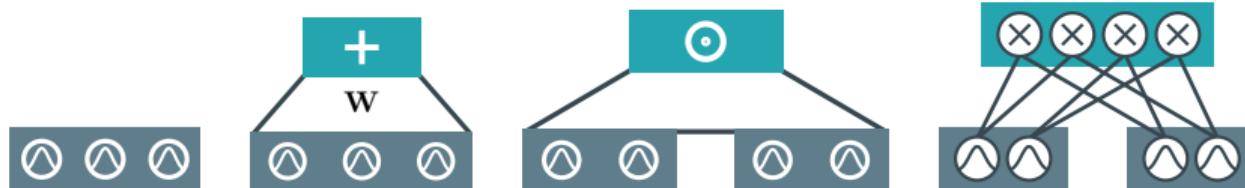


Probabilistic Circuits (PCs)

the layer-wise definition

- I. A set of tractable functions is a circuit layer
- II. A linear projection of a layer is a circuit layer
- III. The product of two layers is a circuit layer

$$c(\mathbf{x}) = \text{vec}(\mathbf{l}(\mathbf{x})\mathbf{r}(\mathbf{x})^\top) \quad // \text{ Kronecker}$$

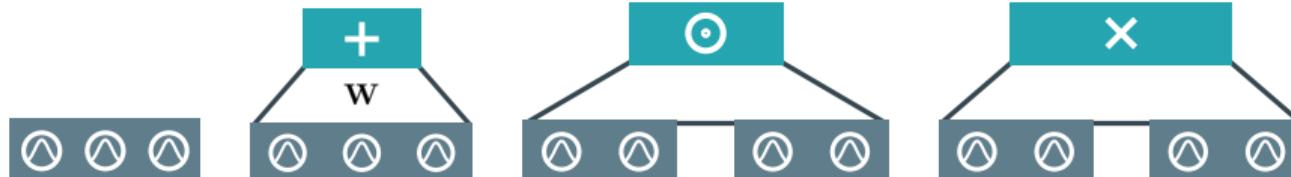


Probabilistic Circuits (PCs)

the layer-wise definition

- I. A set of tractable functions is a circuit layer
- II. A linear projection of a layer is a circuit layer
- III. The product of two layers is a circuit layer

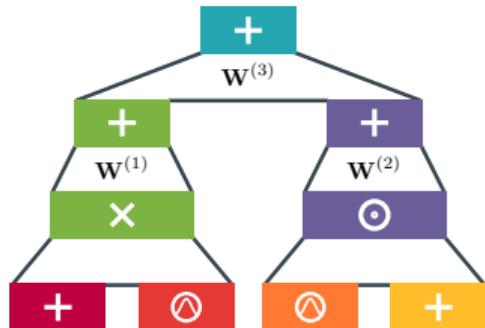
$$c(\mathbf{x}) = \text{vec}(\mathbf{l}(\mathbf{x})\mathbf{r}(\mathbf{x})^\top) \quad // \text{ Kronecker}$$



Probabilistic Circuits (PCs)

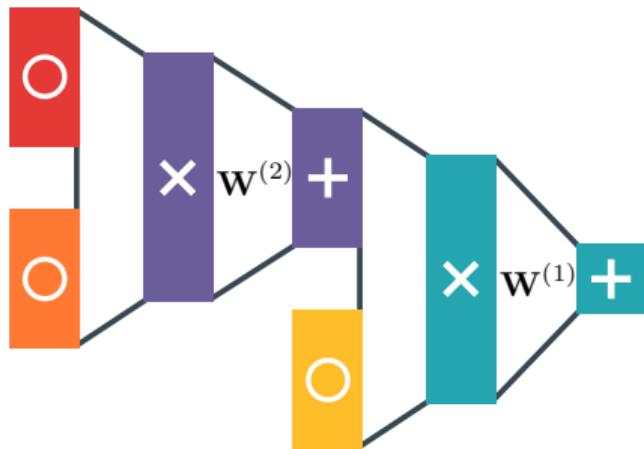
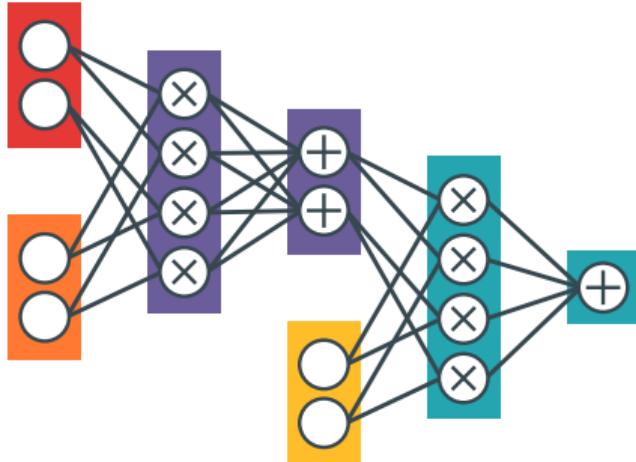
the layer-wise definition

- I. A set of tractable functions is a circuit layer
- II. A linear projection of a layer is a circuit layer
- III. The product of two layers is a circuit layer

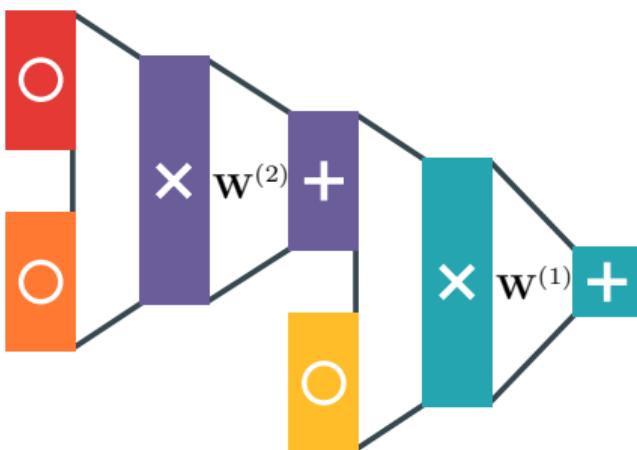
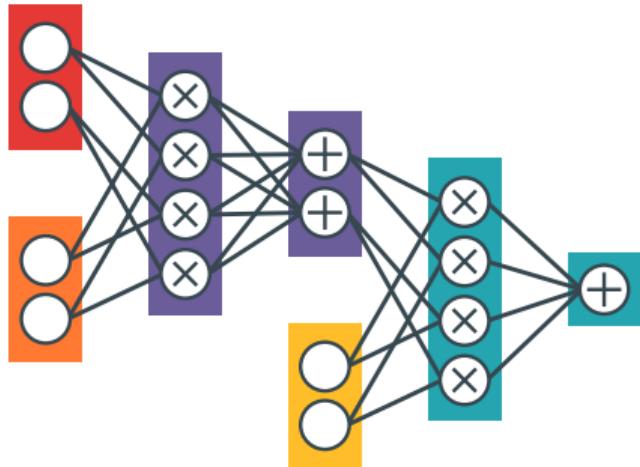


stack layers to build a deep circuit!

Abstracting layers



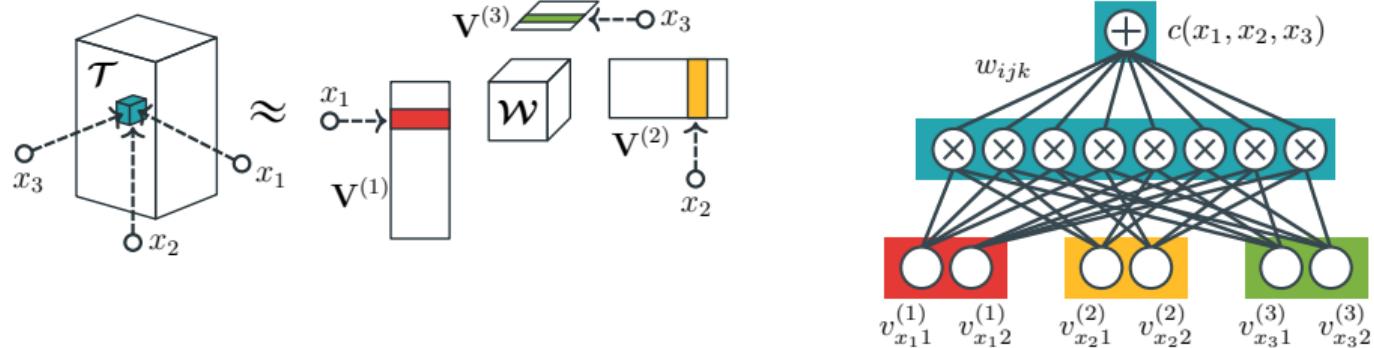
Abstracting layers



combine sum-product layers together to create new layers

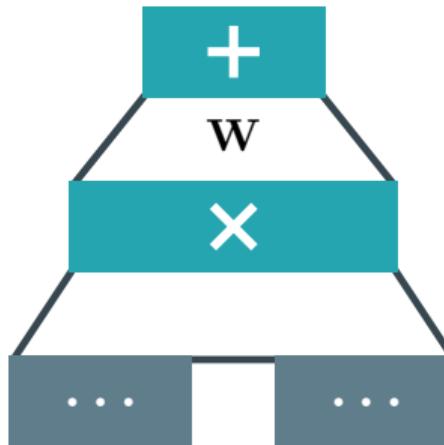
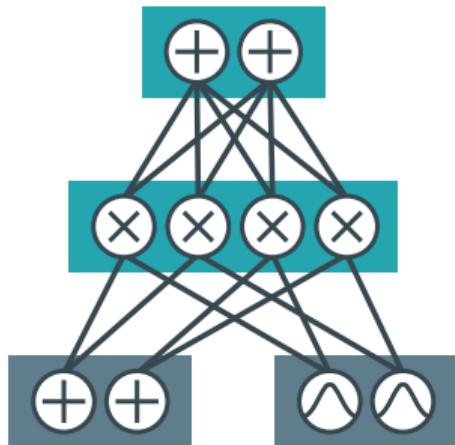
circuits layers

as tensor factorizations



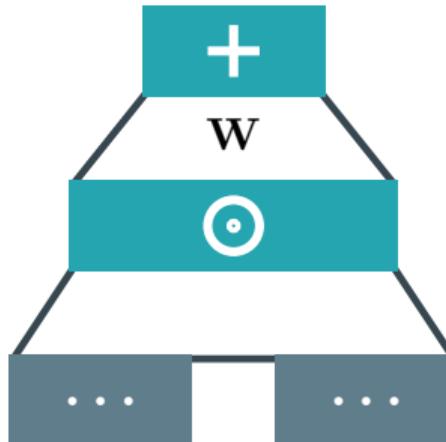
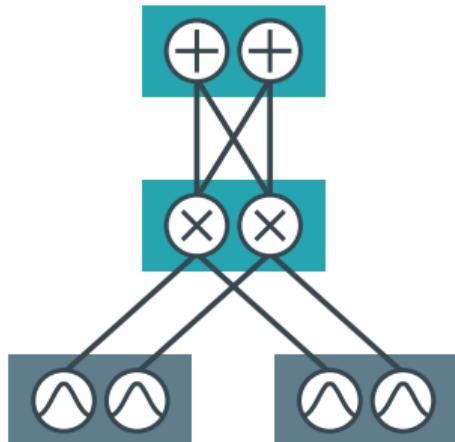
more layers

Tucker decomposition

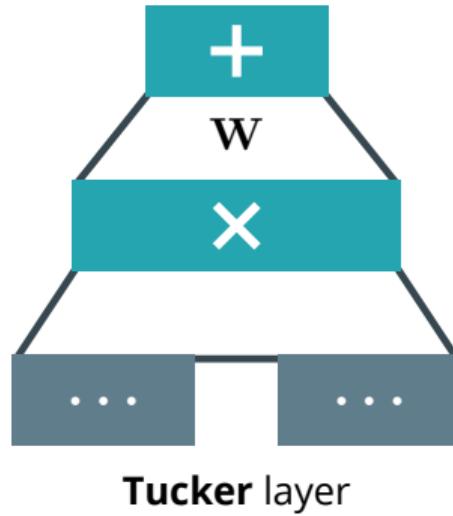
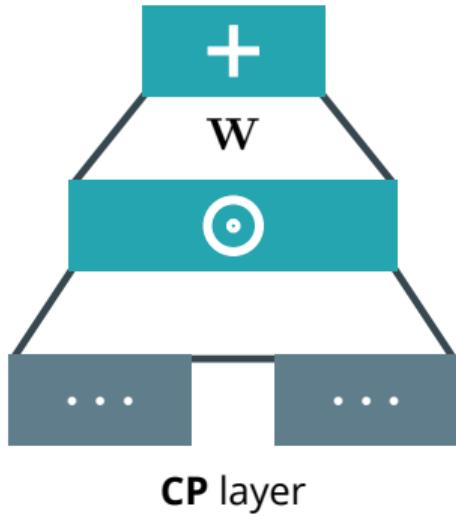


more layers

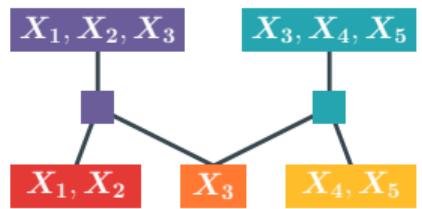
Candecomp Parafac (CP) decomposition



more layers

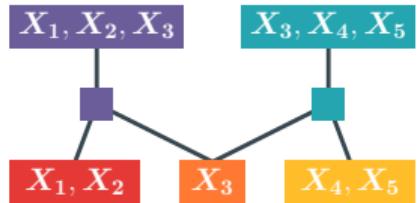


Learning recipe

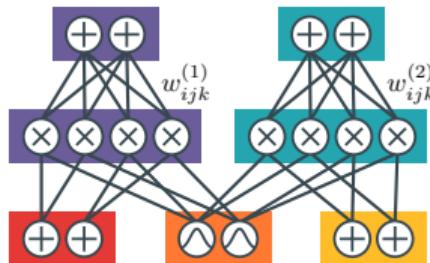


1) Build a *region graph*

Learning recipe



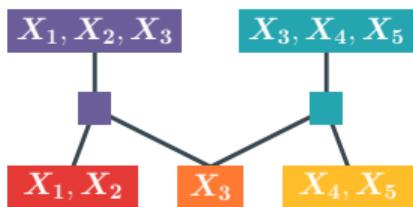
1) Build a *region graph*



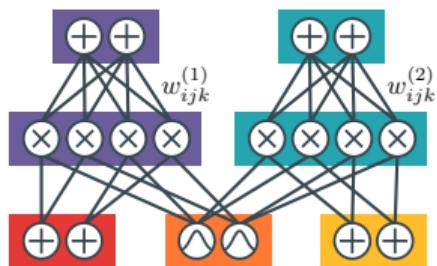
2) Overparameterize

- 2.1) pick a (composite) layer type**
- 2.2) choose how many units per layer**

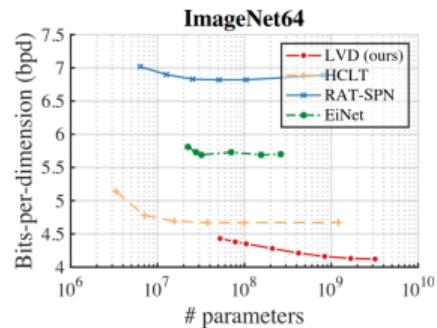
Learning recipe



1) Build a *region graph*

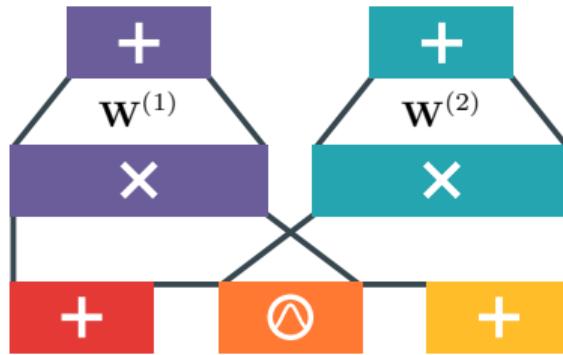


2) Overparameterize



3) Learn parameters

folding layers



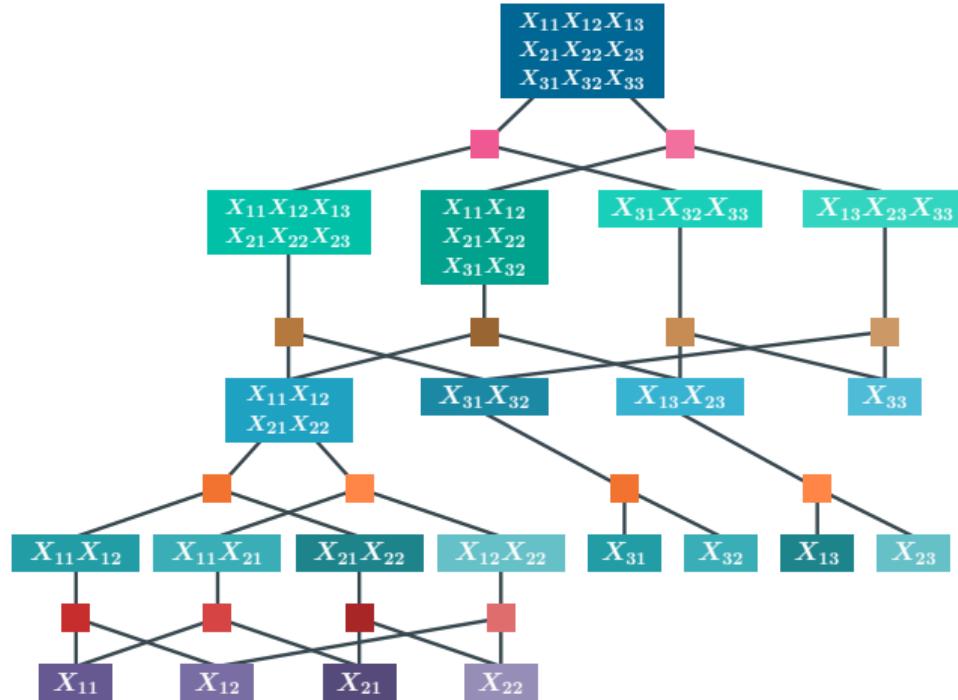
stack layers to increase GPU parallelism

building circuits

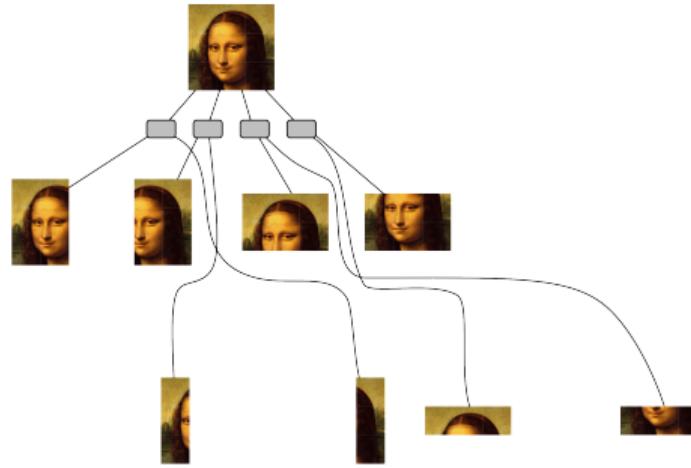
mix&match RGs and layers

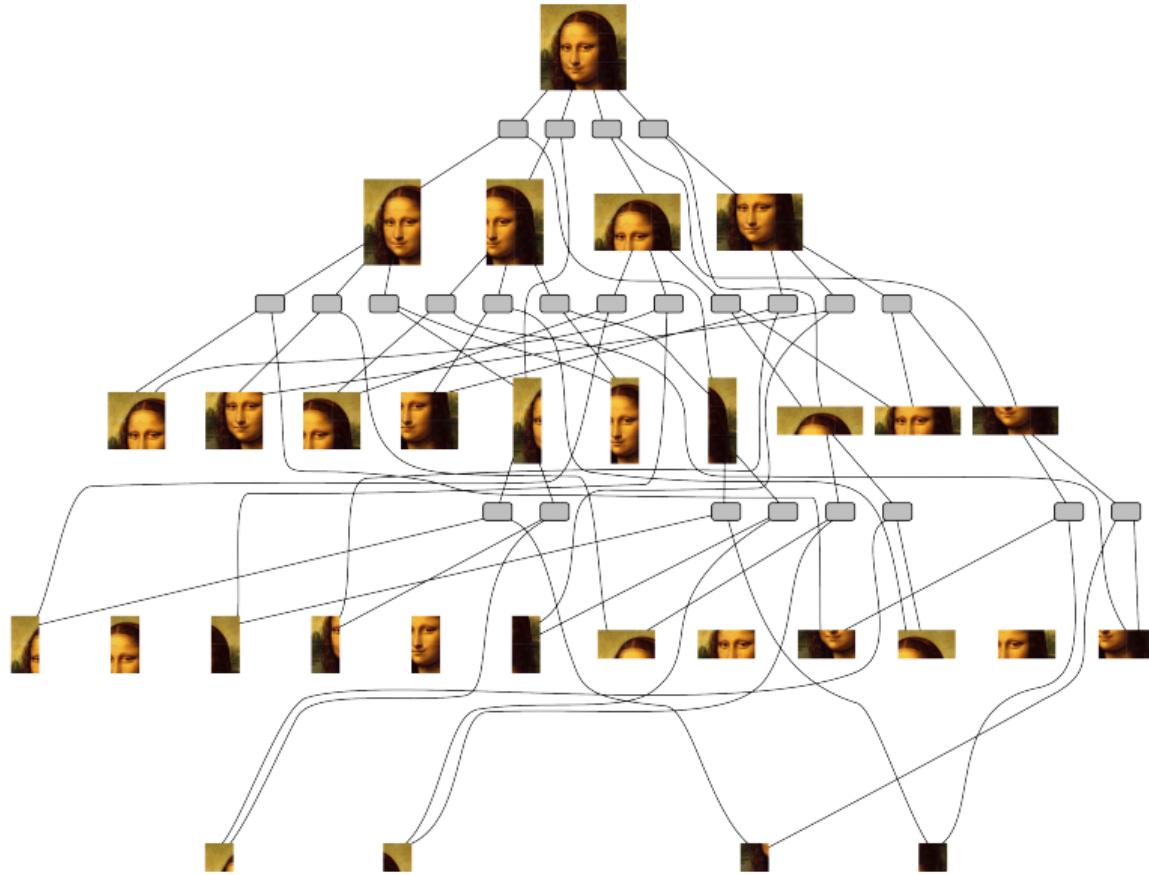
PC ARCHITECTURE	REGION GRAPH	SUM-PRODUCT LAYER	FOLD
Poon&Domingos (Poon & Domingos, 2011)	PD	CP	✗
RAT-SPN (Peharz et al., 2020b)	RND	Tucker	✗
EiNet (Peharz et al., 2020a)	{ RND, PD }	Tucker	✓
HCLT (Liu & Van den Broeck, 2021)	CL	CP	✗
HMM/MPS $_{\mathbb{R} \geq 0}$ (Glasser et al., 2019)	LT	CP	✗
BM (Han et al., 2018)	LT	CP	✗
TTDE (Novikov et al., 2021)	LT	CP	✗
NPC ² (Loconte et al., 2024)	{ LT, RND }	{ CP, Tucker }	✓
TTN (Cheng et al., 2019)	QT-2	Tucker	✗
Mix & Match (our pipeline)	$\left\{ \begin{array}{l} \text{RND, PD, LT,} \\ \text{CL, QG, QT-2, QT-4} \end{array} \right\}$	$\times \left\{ \begin{array}{l} \text{CP, Tucker} \\ \text{CP}^S, \text{CP}^{XS} \end{array} \right\} \cup \left\{ \begin{array}{l} \text{CP, Tucker} \\ \text{CP}^S, \text{CP}^{XS} \end{array} \right\}$	$\times \left\{ \begin{array}{l} \text{✗, ✓} \\ \text{✗, ✓} \end{array} \right\}$

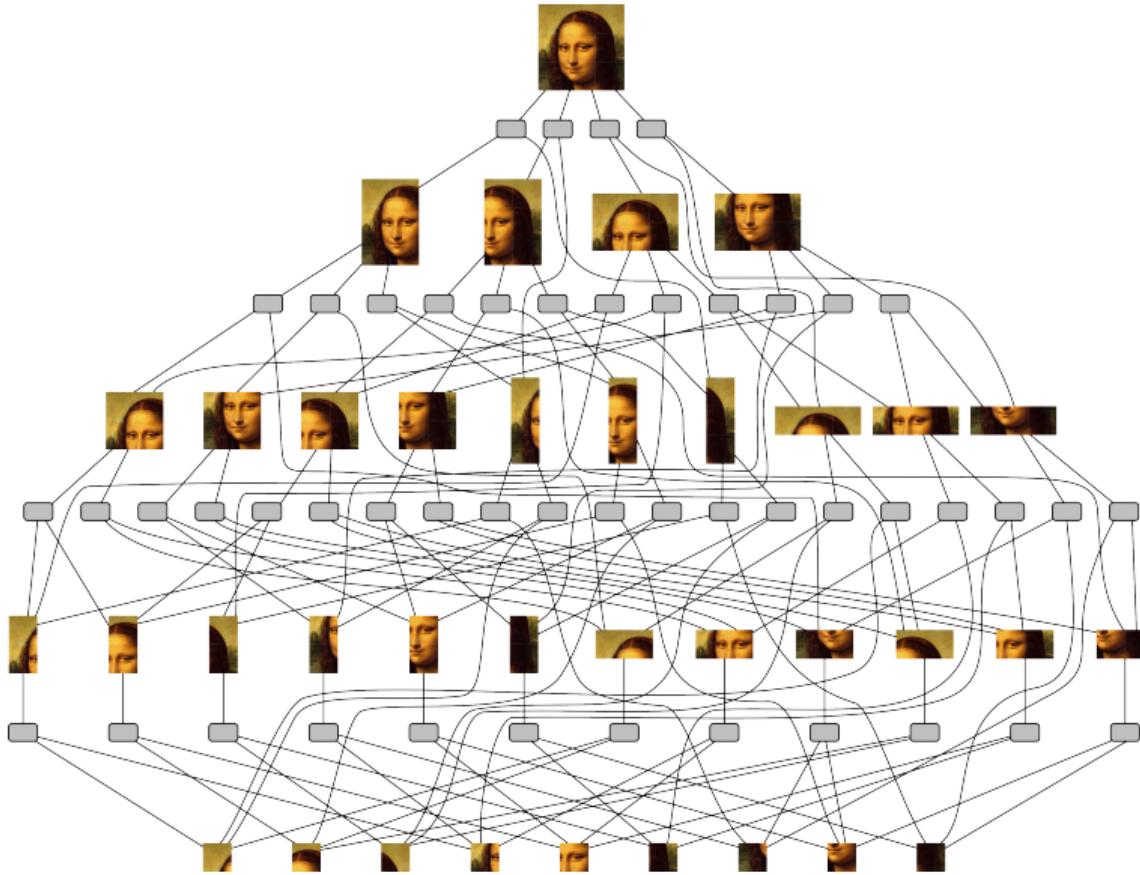
which region graph?









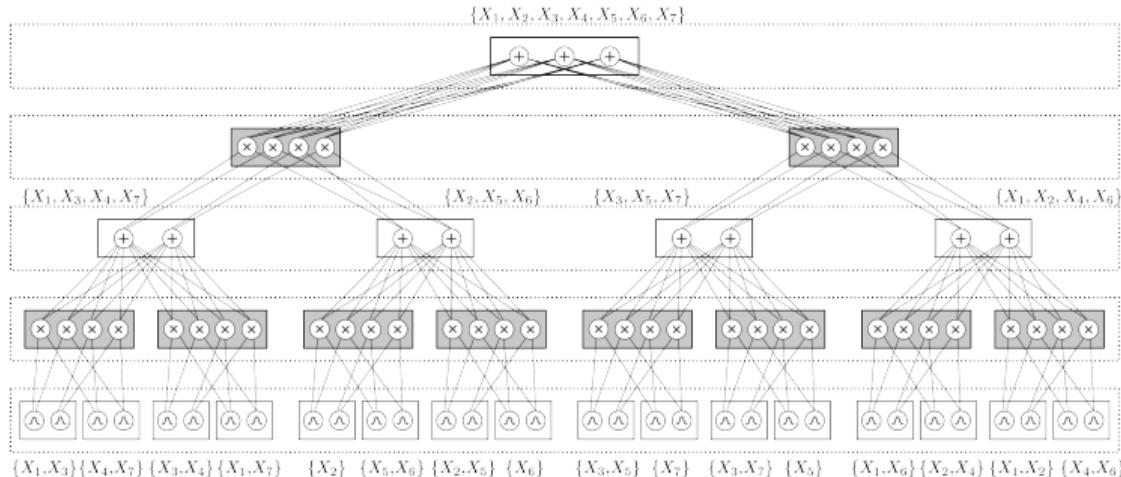


random regions graphs

The “no-learning” option

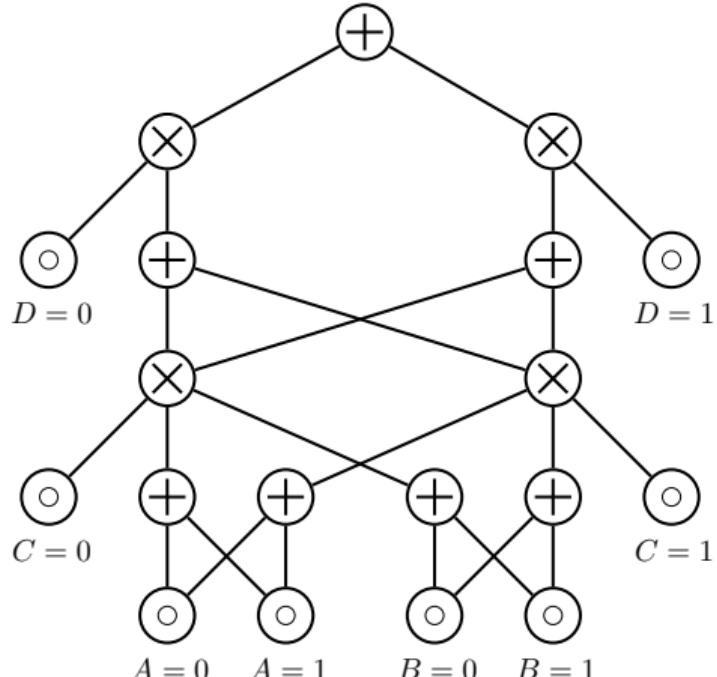
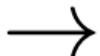
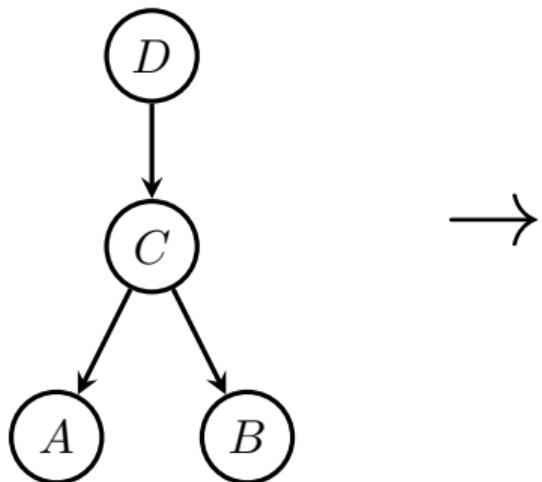
[Peharz et al. 2019a]

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



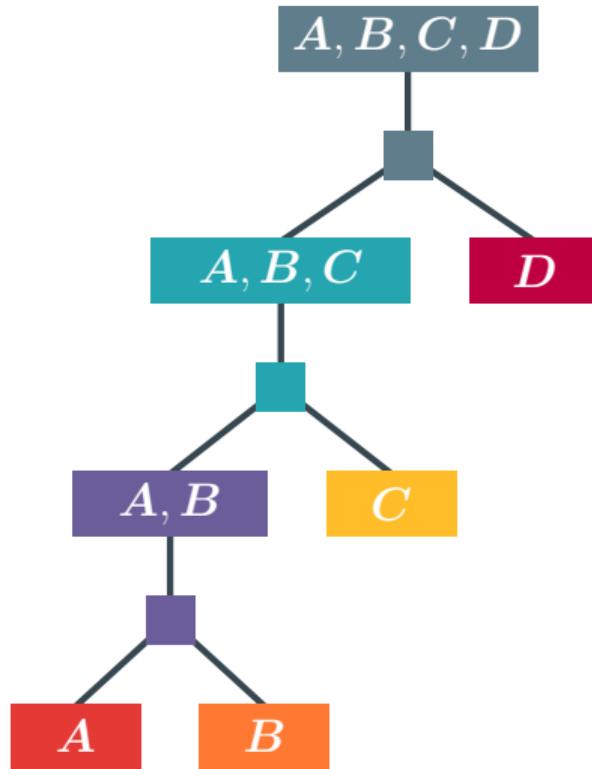
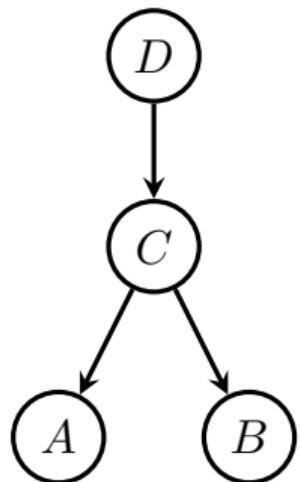
RGs from PGMs

via compilation



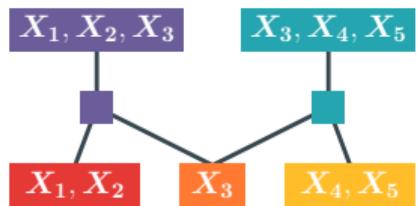
RGs from PGMs

via compilation



learning recipe

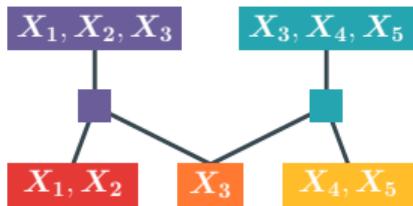
recap



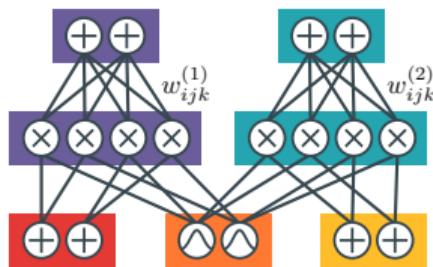
1) Build a ***region graph***

learning recipe

recap



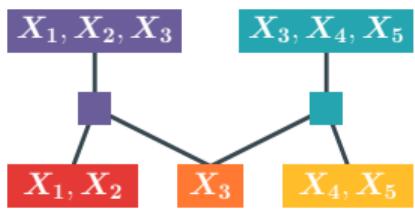
1) Build a *region graph*



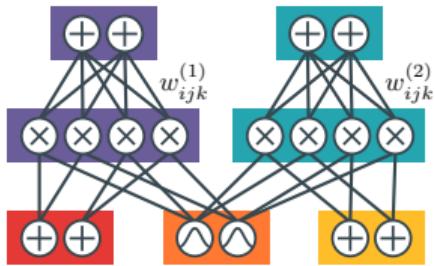
2) Overparameterize

learning recipe

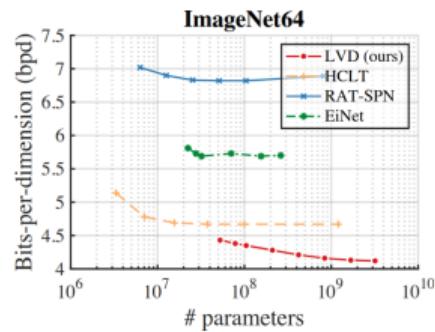
recap



1) Build a *region graph*



2) Overparameterize



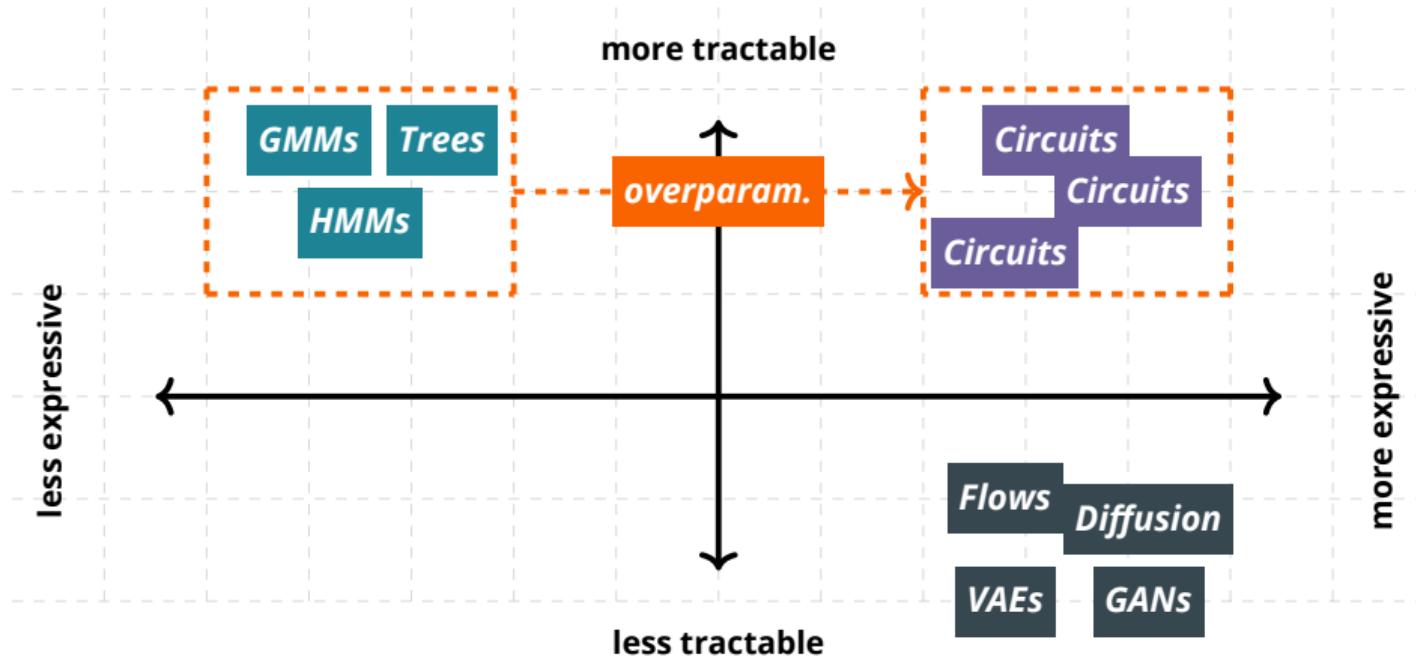
3) Learn parameters

learning probabilistic circuits

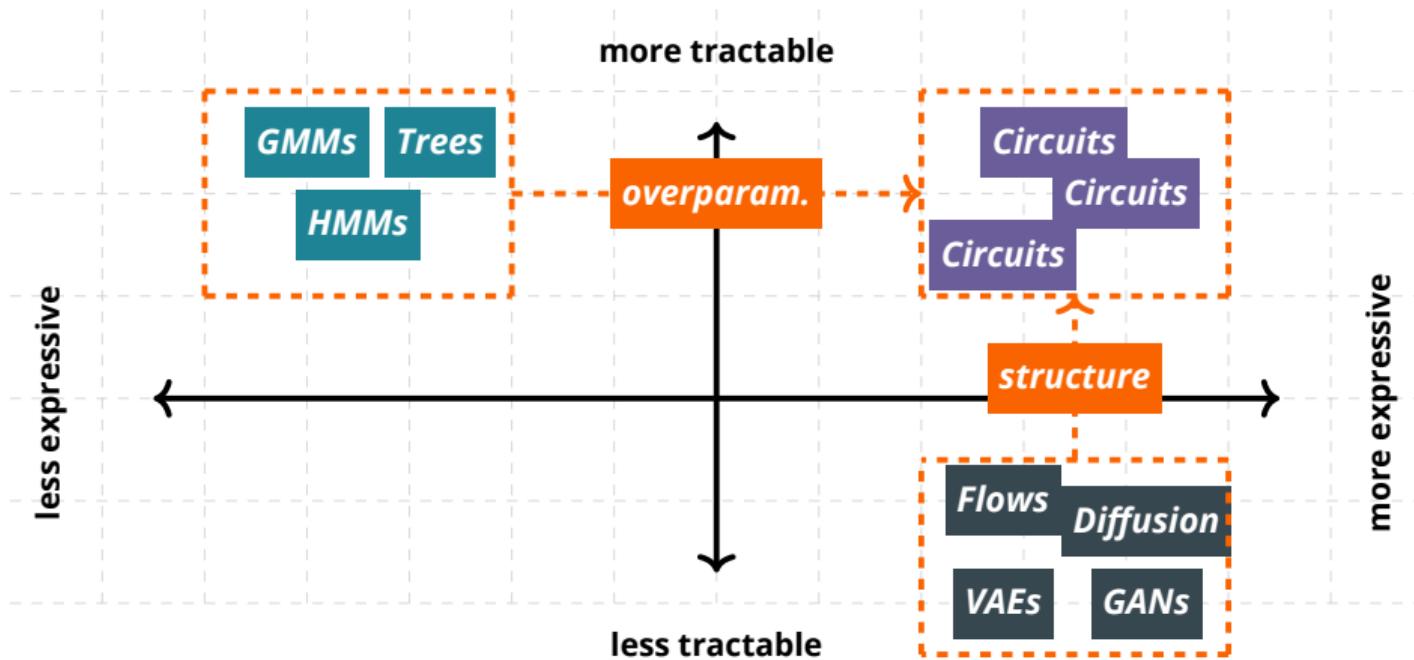
	<i>Parameters</i>	<i>Structure</i>
<i>Generative</i>	<i>deterministic</i> closed-form MLE [Kisa et al. 2014; Peharz et al. 2014]	<i>greedy</i> top-down [Gens et al. 2013; Rahman et al. 2014; Rooshenas et al. 2014] [Vergari et al. 2019]
	<i>non-deterministic</i> EM [Poon et al. 2011; Peharz 2015; Zhao et al. 2016b] SGD [Sharir et al. 2016; Peharz et al. 2019b] Bayesian [Jaini et al. 2016; Rashwan et al. 2016] [Zhao et al. 2016a; Trapp et al. 2019; Vergari et al. 2019]	<i>hill climb</i> [Dennis et al. 1967] [Dang et al. 2022] a single pipeline [al. 2019]
<i>Discriminative</i>	?	?

learning probabilistic circuits

	<i>Parameters</i>	<i>Structure</i>
Generative	deterministic closed-form MLE [Kisa et al. 2014; Peharz et al. 2014]	greedy top-down [Gens et al. 2013; Rahman et al. 2014; Rooshenas et al. 2014] [Vergari et al. 2015] bottom-up [Peharz et al. 2013]
	non-deterministic EM [Poon et al. 2011; Peharz 2015] [Zhao et al. 2016b] SGD [Sharir et al. 2016; Peharz et al. 2019b] Bayesian [Jaini et al. 2016; Rashwan et al. 2016] [Zhao et al. 2016a; Trapp et al. 2019; Vergari et al. 2019]	hill climbing [Lowd et al. 2008, 2013; Peharz et al. 2014] [Dennis et al. 2015; Liang et al. 2017; Galindez Olascoaga et al. 2019] [Dang et al. 2022] random RAT-SPNs [Peharz et al. 2019b] XCNet [Di Mauro et al. 2017]
Discriminative	deterministic convex-opt MLE [Liang et al. 2019]	greedy top-down [Shao et al. 2020]
	non-deterministic EM [Rashwan et al. 2018] SGD [Gens et al. 2012; Sharir et al. 2016] [Peharz et al. 2019b]	hill climbing [Rooshenas et al. 2016; Liang et al. 2019]



then make it more expressive!



impose structure!

References I

- ⊕ Darwiche, Adnan and Pierre Marquis (2002). "A knowledge compilation map". In: Journal of Artificial Intelligence Research 17, pp. 229–264.
- ⊕ Darwiche, Adnan (2003). "A Differential Approach to Inference in Bayesian Networks". In: JACM.
- ⊕ Lowd, Daniel and Pedro Domingos (2008). "Learning Arithmetic Circuits". In: Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence. UAI'08. Helsinki, Finland: AUAI Press, pp. 383–392. ISBN: 0-9749039-4-9. URL: <http://dl.acm.org/citation.cfm?id=3023476.3023522>.
- ⊕ Poon, Hoifung and Pedro Domingos (2011). "Sum-Product Networks: a New Deep Architecture". In: UAI 2011.
- ⊕ Gens, Robert and Pedro Domingos (2012). "Discriminative Learning of Sum-Product Networks". In: Advances in Neural Information Processing Systems 25, pp. 3239–3247.
- ⊕ — (2013). "Learning the Structure of Sum-Product Networks". In: Proceedings of the ICML 2013, pp. 873–880.
- ⊕ Lowd, Daniel and Amirmohammad Rooshenas (2013). "Learning Markov Networks With Arithmetic Circuits". In: Proceedings of the 16th International Conference on Artificial Intelligence and Statistics. Vol. 31. JMLR Workshop Proceedings, pp. 406–414.
- ⊕ Peharz, Robert, Bernhard Geiger, and Franz Pernkopf (2013). "Greedy Part-Wise Learning of Sum-Product Networks". In: ECML-PKDD 2013.
- ⊕ Kisa, Doga, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche (July 2014). "Probabilistic sentential decision diagrams". In: Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR). Vienna, Austria. URL: <http://starai.cs.ucla.edu/papers/KisaKR14.pdf>.
- ⊕ Peharz, Robert, Robert Gens, and Pedro Domingos (2014). "Learning Selective Sum-Product Networks". In: Workshop on Learning Tractable Probabilistic Models. LTPM.

References II

- ⊕ Rahman, Tahrima, Prasanna Kothalkar, and Vibhav Gogate (2014). "Cutset Networks: A Simple, Tractable, and Scalable Approach for Improving the Accuracy of Chow-Liu Trees". In: Machine Learning and Knowledge Discovery in Databases. Vol. 8725. LNCS. Springer, pp. 630–645.
- ⊕ Rooshenas, Amirmohammad and Daniel Lowd (2014). "Learning Sum-Product Networks with Direct and Indirect Variable Interactions". In: Proceedings of ICML 2014.
- ⊕ Dennis, Aaron and Dan Ventura (2015). "Greedy Structure Search for Sum-product Networks". In: IJCAI'15. Buenos Aires, Argentina: AAAI Press, pp. 932–938. ISBN: 978-1-57735-738-4.
- ⊕ Peharz, Robert (2015). "Foundations of Sum-Product Networks for Probabilistic Modeling". PhD thesis. Graz University of Technology, SPSC.
- ⊕ Vergari, Antonio, Nicola Di Mauro, and Floriana Esposito (2015). "Simplifying, Regularizing and Strengthening Sum-Product Network Structure Learning". In: ECML-PKDD 2015.
- ⊕ Friesen, Abram L and Pedro Domingos (2016). "Submodular Sum-product Networks for Scene Understanding". In.
- ⊕ Jaini, Priyank, Abdullah Rashwan, Han Zhao, Yue Liu, Ershad Banijamali, Zhitang Chen, and Pascal Poupart (2016). "Online Algorithms for Sum-Product Networks with Continuous Variables". In: Probabilistic Graphical Models - Eighth International Conference, PGM 2016, Lugano, Switzerland, September 6-9, 2016. Proceedings, pp. 228–239. URL: <http://jmlr.org/proceedings/v52/jaini16.html>.
- ⊕ Peharz, Robert, Robert Gens, Franz Pernkopf, and Pedro M. Domingos (2016). "On the Latent Variable Interpretation in Sum-Product Networks". In: IEEE Transactions on Pattern Analysis and Machine Intelligence PP, Issue 99. URL: <http://arxiv.org/abs/1601.06180>.
- ⊕ Rashwan, Abdullah, Han Zhao, and Pascal Poupart (2016). "Online and Distributed Bayesian Moment Matching for Parameter Learning in Sum-Product Networks". In: Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, pp. 1469–1477.

References III

- ⊕ Rooshenas, Amirmohammad and Daniel Lowd (2016). "Discriminative Structure Learning of Arithmetic Circuits". In: Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, pp. 1506–1514.
- ⊕ Sharir, Or, Ronen Tamari, Nadav Cohen, and Amnon Shashua (2016). "Tractable generative convolutional arithmetic circuits". In: arXiv preprint arXiv:1610.04167.
- ⊕ Yuan, Zehuan, Hao Wang, Limin Wang, Tong Lu, Shivakumara Palaiahnakote, and Chew Lim Tan (2016). "Modeling spatial layout for scene image understanding via a novel multiscale sum-product network". In: Expert Systems with Applications 63, pp. 231–240.
- ⊕ Zhao, Han, Tameem Adel, Geoff Gordon, and Brandon Amos (2016a). "Collapsed Variational Inference for Sum-Product Networks". In: In Proceedings of the 33rd International Conference on Machine Learning, Vol. 48.
- ⊕ Zhao, Han, Pascal Poupart, and Geoffrey J Gordon (2016b). "A Unified Approach for Learning the Parameters of Sum-Product Networks". In: Advances in Neural Information Processing Systems 29, Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., pp. 433–441.
- ⊕ Di Mauro, Nicola, Antonio Vergari, Teresa M. A. Basile, and Floriana Esposito (2017). "Fast and Accurate Density Estimation with Extremely Randomized Cutset Networks". In: ECML-PKDD 2017.
- ⊕ Liang, Yitao, Jessa Bekker, and Guy Van den Broeck (2017). "Learning the structure of probabilistic sentential decision diagrams". In: Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI).
- ⊕ Rashwan, Abdullah, Pascal Poupart, and Chen Zhitang (2018). "Discriminative Training of Sum-Product Networks by Extended Baum-Welch". In: International Conference on Probabilistic Graphical Models, pp. 356–367.

References IV

- ⊕ Galindez Olascoaga, Laura I, Wannes Meert, Nimish Shah, Marian Verhelst, and Guy Van den Broeck (2019). "Towards hardware-aware tractable learning of probabilistic models". In: [Advances in Neural Information Processing Systems 32](#).
- ⊕ Liang, Yitao and Guy Van den Broeck (2019). "Learning Logistic Circuits". In: [Proceedings of the 33rd Conference on Artificial Intelligence \(AAAI\)](#).
- ⊕ Peharz, Robert, Antonio Vergari, Karl Stelzner, Alejandro Molina, Xiaoting Shao, Martin Trapp, Kristian Kersting, and Zoubin Ghahramani (2019a). "Random sum-product networks: A simple but effective approach to probabilistic deep learning". In: [UAI](#).
- ⊕ Peharz, Robert, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Xiaoting Shao, Kristian Kersting, and Zoubin Ghahramani (2019b). "Random Sum-Product Networks: A Simple and Effective Approach to Probabilistic Deep Learning". In: [Uncertainty in Artificial Intelligence](#).
- ⊕ Trapp, Martin, Robert Peharz, Hong Ge, Franz Pernkopf, and Zoubin Ghahramani (2019). "Bayesian Learning of Sum-Product Networks". In: [Advances in neural information processing systems \(NeurIPS\)](#).
- ⊕ Vergari, Antonio, Alejandro Molina, Robert Peharz, Zoubin Ghahramani, Kristian Kersting, and Isabel Valera (2019). "Automatic Bayesian density analysis". In: [AAAI 33](#), pp. 5207–5215.
- ⊕ Peharz, Robert, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van den Broeck, Kristian Kersting, and Zoubin Ghahramani (2020). "Einsum Networks: Fast and Scalable Learning of Tractable Probabilistic Circuits". In: [ICML](#).
- ⊕ Shao, Xiaoting, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, and Kristian Kersting (2020). "Conditional Sum-Product Networks: Imposing Structure on Deep Probabilistic Architectures". In: [PGM](#).
- ⊕ Shih, Andy and Stefano Ermon (2020). "Probabilistic Circuits for Variational Inference in Discrete Graphical Models". In: [NeurIPS](#).

References V

- ⊕ Vergari, Antonio, YooJung Choi, Anji Liu, Stefano Teso, and Guy Van den Broeck (2021). "A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference". In: NeurIPS. arXiv: 2102.06137 [stat.ML].
- ⊕ Dang, Meihua, Antonio Vergari, and Guy Van den Broeck (2022). "Strudel: A fast and accurate learner of structured-decomposable probabilistic circuits". In: International Journal of Approximate Reasoning 140, pp. 92–115.
- ⊕ Liu, Anji, Honghua Zhang, and Guy Van den Broeck (2022). "Scaling Up Probabilistic Circuits by Latent Variable Distillation". In: The Eleventh International Conference on Learning Representations.
- ⊕ Mari, Antonio, Gennaro Vessio, and Antonio Vergari (2023). "Unifying and Understanding Overparameterized Circuit Representations via Low-Rank Tensor Decompositions". In: 6th Workshop on Tractable Probabilistic Modeling.
- ⊕ Gala, Gennaro, Cassio de Campos, Antonio Vergari, and Erik Quaeghebeur (2024). Scaling Continuous Latent Variable Models as Probabilistic Integral Circuits. arXiv: 2406.06494.
- ⊕ Liu, Anji, Mathias Niepert, and Guy Van den Broeck (2024). "Image Inpainting via Tractable Steering of Diffusion Models". In: ICLR.
- ⊕ Mari, Antonio, Lorenzo Loconte, Gennaro Gala, Erik Quaeghebeur, Gennaro Vessio, and Antonio Vergari (2024). "What is the Relationship between Circuits and Tensor Factorizations (and How Can We Exploit it)?". In: Under submission.