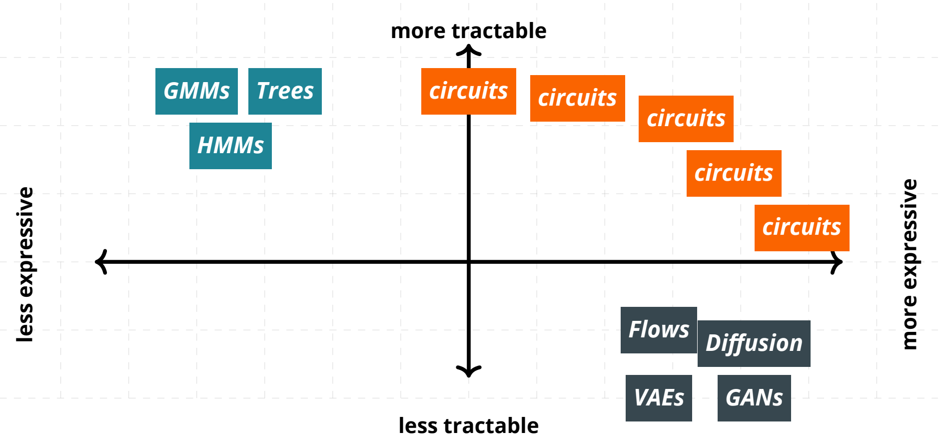
advanced
probabilistic modeling & reasoning

antonio vergari (he/him)

@tetraduzione

18th Oct 2024 - **PICS PhD School** *Copenhagen*



trade-off tractability vs expressiveness

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

q_1 $\int p(\mathbf{x}_o, \mathbf{x}_m) d\mathbf{X}_m$

(missing values)

q_2 $\mathbb{E}_{\mathbf{x}_c \sim p(\mathbf{X}_c | X_s = 0)} [f_0(\mathbf{x}_c)] - \mathbb{E}_{\mathbf{x}_c \sim p(\mathbf{X}_c | X_s = 1)} [f_1(\mathbf{x}_c)]$

(fairness)

q_3 $\mathbb{E}_{\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_D)} [f(\mathbf{x} + \mathbf{e})]$

(adversarial robust.)

...in the language of probabilities

Reasoning about ML models

q_1 $\int p(\mathbf{x}_o, \mathbf{x}_m) d\mathbf{X}_m$

(missing values)

q_2 $\mathbb{E}_{\mathbf{x}_c \sim p(\mathbf{X}_c | X_s = 0)} \left[f_0(\mathbf{x}_c) \right] - \mathbb{E}_{\mathbf{x}_c \sim p(\mathbf{X}_c | X_s = 1)} \left[f_1(\mathbf{x}_c) \right]$

(fairness)

q_3 $\mathbb{E}_{\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_D)} \left[f(\mathbf{x} + \mathbf{e}) \right]$

(adversarial robust.)

it is crucial we compute them exactly and in polytime!

Reasoning about ML models

q_1 $\int p(\mathbf{x}_o, \mathbf{x}_m) d\mathbf{X}_m$
(missing values)

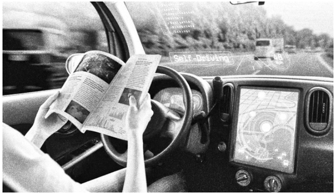
q_2 $\mathbb{E}_{\mathbf{x}_c \sim p(\mathbf{X}_c | X_s=0)} \left[f_0(\mathbf{x}_c) \right] - \mathbb{E}_{\mathbf{x}_c \sim p(\mathbf{X}_c | X_s=1)} \left[f_1(\mathbf{x}_c) \right]$
(fairness)

q_3 $\mathbb{E}_{\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_D)} \left[f(\mathbf{x} + \mathbf{e}) \right]$
(adversarial robust.)

it is crucial we compute them tractably!

Which structural properties

for complex reasoning



smooth + decomposable

Which structural properties

for complex reasoning

smooth + decomposable

???????

decomposable

Which structural properties

for complex reasoning

smooth + decomposable

compatibility

decomposable

The problem!

queries (behaviors)

model classes	q_1: $\mathbb{E}_{\mathbf{x}_m}\left[f(\mathbf{x}_o, \mathbf{x}_m)\right]$	q_2: fairness queries	q_3: $\mathbb{E}_{\mathbf{e}\sim\mathcal{N}}\left[f(\mathbf{x}+\mathbf{e})\right]$	q_4: robustness queries	q_5: $\mathbb{E}_{\mathbf{x}_m}\left[f^{\alpha}(\mathbf{x}_o, \mathbf{x}_m)\right]$	…
HMMs M_1	✓	?	✓	✓	✗	…
decision trees M_2	✓	✓	✓	✓	✓	…
neural nets w/ ReLUs M_3	✗	?	?	✓	✗	…
tensor factorizations M_4	✓	?	✓	✓	?	…
transformers M_5	?	?	?	?	?	…
…	…	…	…	…	…	…

model classes

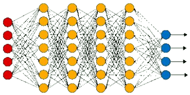
✓ reliable & efficient algo

✗ intractable

? do not know

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

behaviors
to be inspected



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

???

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

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

How can we solve this *engineering bottleneck*?

queries (behaviors)

		q_1 $\mathbb{E}_{\mathbf{x}_m}[f(\mathbf{x}_o, \mathbf{x}_m)]$	q_2 fairness queries	q_3 $\mathbb{E}_{\mathbf{e} \sim \mathcal{N}}[f(\mathbf{x}+\mathbf{e})]$	q_4 robustness queries	q_5 $\mathbb{E}_{\mathbf{x}_m}[f^{\alpha}(\mathbf{x}_o, \mathbf{x}_m)]$	...
HMMs	M_1	✓	?	✓	✓	✗	...
decision trees	M_2	✓	✓	✓	✓	✓	...
neural nets w/ ReLUs	M_3	✗	?	?	✓	✗	...
tensor factorizations	M_4	✓	?	✓	✓	?	...
transformers	M_5	?	?	?	?	?	...
	...	...	...	...	...	...	...

model classes

✓ reliable & efficient algo

✗ intractable

? do not know

queries (behaviors)

		q_1 $\mathbb{E}_{\mathbf{x}_m}[f(\mathbf{x}_o, \mathbf{x}_m)]$	q_2 fairness queries	q_3 $\mathbb{E}_{\mathbf{e}\sim\mathcal{N}}[f(\mathbf{x}+\mathbf{e})]$	q_4 robustness queries	q_5 $\mathbb{E}_{\mathbf{x}_m}[f^{\alpha}(\mathbf{x}_o, \mathbf{x}_m)]$	...
smoothness	**SMO**	✓	✓	✓	✓	-	...
scope-decomposability	**DEC**	-	✓	✓	-	✓	...
supp-decomposability	**DET**	-	-	-	✓	-	...
compatibility	**CMP**	✓	-	-	✓	-	...

properties of a unified abstraction (circuits)

✓ necess. conditions for reliability and efficiency

atomic queries

	$\times$ ($p \times q$)	$+$ ($p + q$)	$\mathsf{pow}_{\mathbb{N}}$ (p^n)	$\mathsf{pow}_{\mathbb{R}}$ (p^α)	$/$ (p/q)	log ($\log p$)	exp ($\exp p$)
smoothness **SMO**	✓	✓	✓	✓	✓	✓	✓
scope-decomposability **DEC**	-	-	-	✓	-	-	-
supp-decomposability **DET**	-	-	-	✓	✓	✓	-
compatibility **CMP**	-	✓	✓	-	✓	-	-

properties of a unified abstraction (circuits)

✓ 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?"*

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

*"What's the expected **variance**?"*

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

(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 expected **variance**?"*

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

*"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 $/$

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

A language for queries

Integral expressions that can be formed by composing these operators

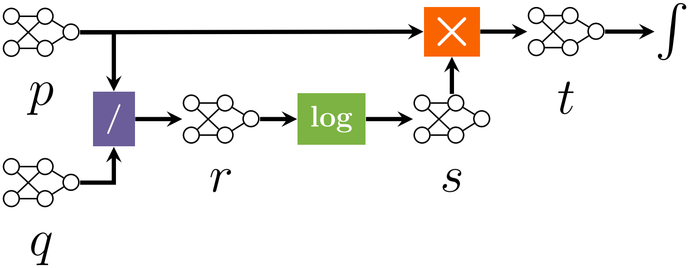
$+$, $\times$, **pow**, **log**, **exp** and $/$

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

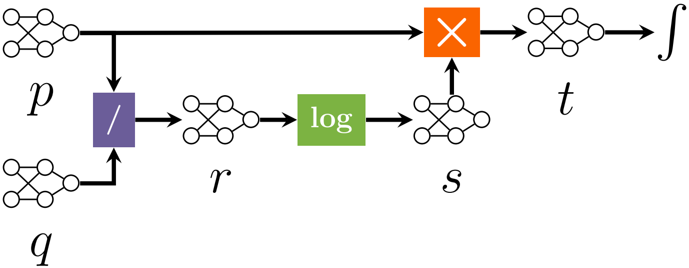
Represented as ***higher-order computational graphs***—pipelines—operating over circuits!

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

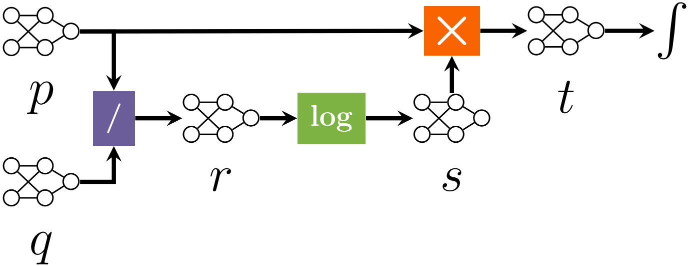
$$\mathbb{KLD}(p \mid\mid q) = \int_{\mathsf{val}(\mathbf{X})} p(\mathbf{x}) \times \log\left(p(\mathbf{x})/q(\mathbf{x})\right) \, d\mathbf{X}$$



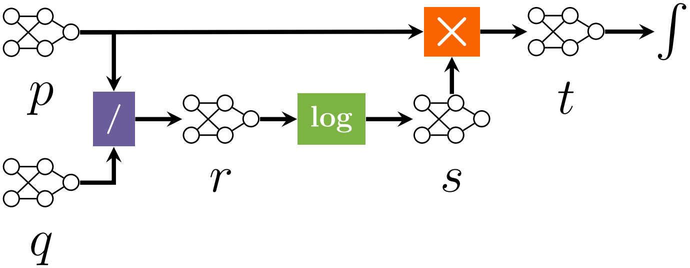
$$\mathbb{KLD}(p \,||\, q) = \int_{\mathsf{val}(\mathbf{X})} p(\mathbf{x}) \times \log\left(p(\mathbf{x}) \boxed{/} q(\mathbf{x})\right) \, d\mathbf{X}$$



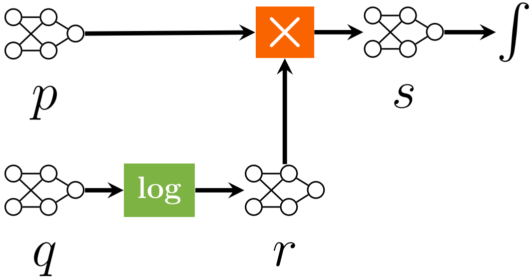
$$\mathbb{KLD}(p \mid\mid q) = \int_{\mathsf{val}(\mathbf{X})} p(\mathbf{x}) \times \boxed{\mathbf{log}} \left(p(\mathbf{x})/q(\mathbf{x})\right) \, d\mathbf{X}$$



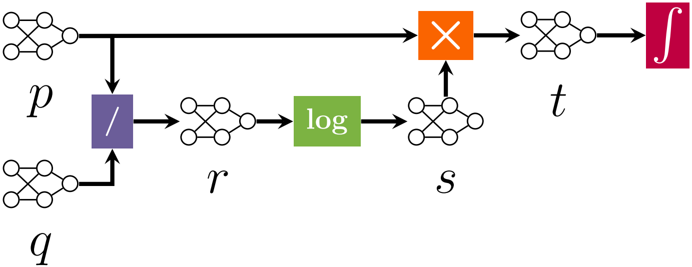
$$\mathbb{KLD}(p \mid\mid q) = \int_{\mathsf{val}(\mathbf{X})} p(\mathbf{x}) \boxed{\times} \log\left(p(\mathbf{x})/q(\mathbf{x})\right)\ d\mathbf{X}$$



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

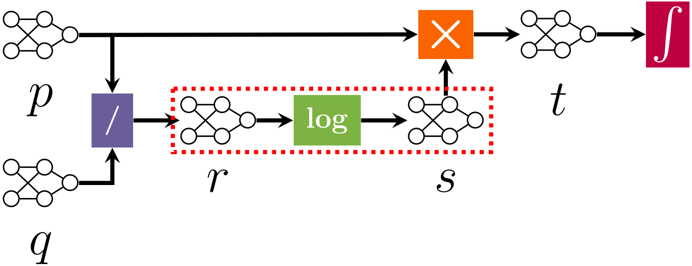


$$\int p(\mathbf{x}) \times \log \left(p(\mathbf{x}) / q(\mathbf{x}) \right) \; d\mathbf{X}$$



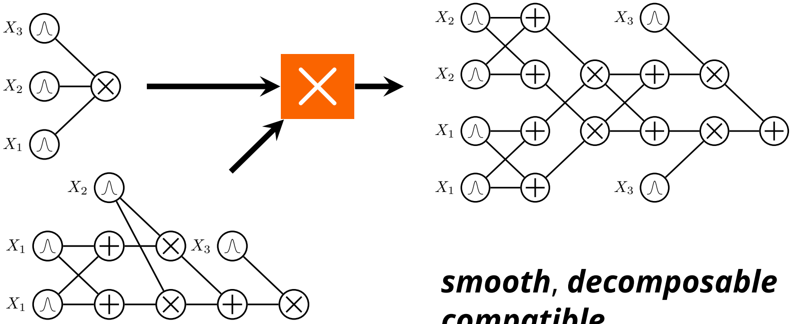
build a LEGO-like query calculus...

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



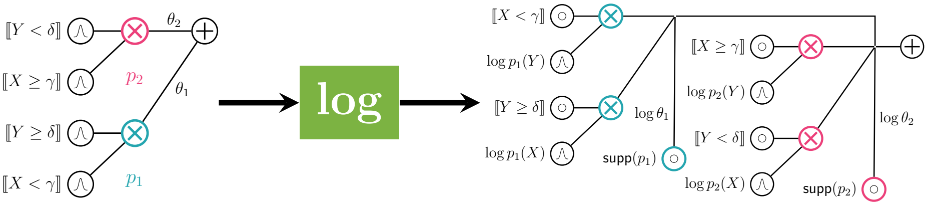
build a LEGO-like query calculus...

Tractable operators



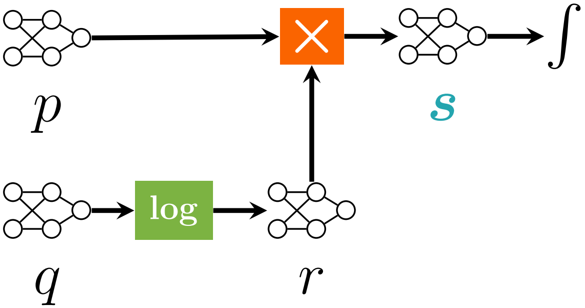
smooth, decomposable
compatible

Tractable operators

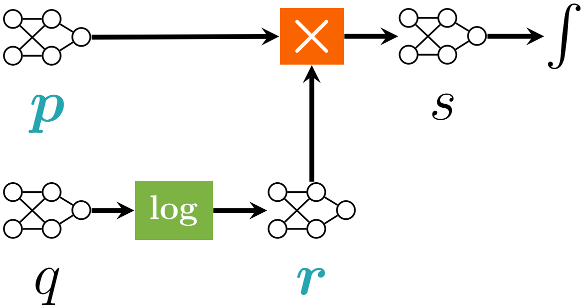


smooth*, *decomposable
deterministic

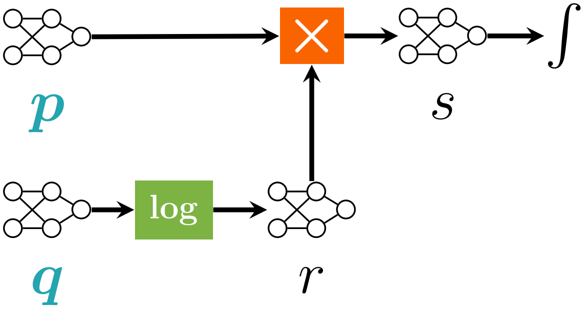
smooth*, *decomposable



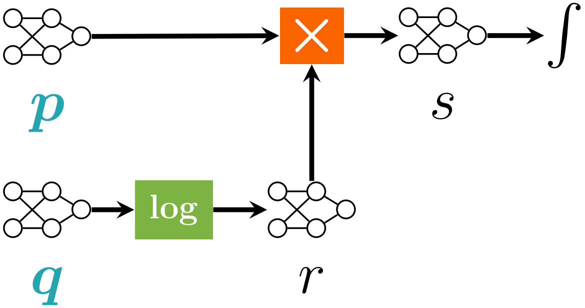
To perform tractable integration we need $\boldsymbol{s}$ to be ***smooth and decomposable…***



hence we need $\boldsymbol{p}$ and $\boldsymbol{r}$ to be smooth, decomposable and ***compatible***...



therefore $\boldsymbol{q}$ must be smooth, decomposable and ***deterministic***...



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

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

compositionally *derive the tractability of many more queries*

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

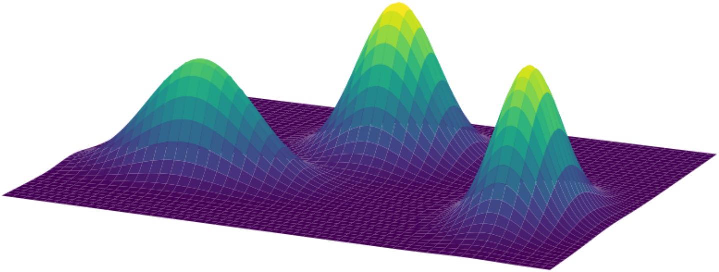
*and **prove hardness** when some input properties are not satisfied*

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

compositional inference

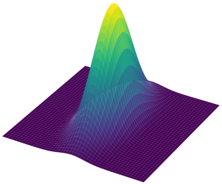
```python
from cirkit.symbolic.functional import integrate, multiply

def expectation(p, f):
    i = multiply(p, f)
    return integrate(i)

def squared_loss(p, q): # \int (p - q)^2
    p2 = multiply(p, p)
    q2 = multiply(q, q)
    pq = multiply(p, q)
    return integrate(p2) + integrate(q2) -2*integrate(pq)
```

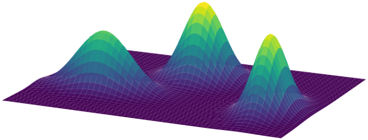
so far…



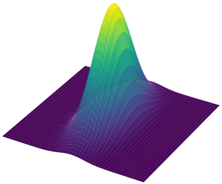
oh mixtures, you're so fine you blow my mind!

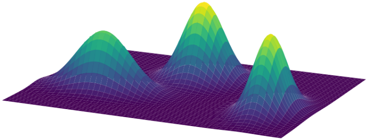
image taken from Hao Tang's course on ASR





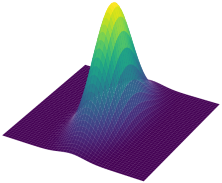
$$p(\mathbf{X}) \rightarrow \sum_{i=1}^{K} w_i p_i(\mathbf{X})$$



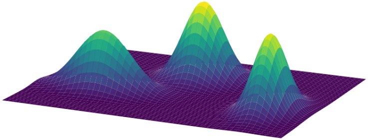


$$p(\mathbf{X}) \rightarrow \sum_{i=1}^{K} w_i p_i(\mathbf{X})$$

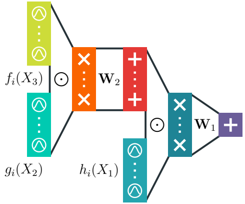
“if someone publishes a paper on model A, there will be a paper about mixtures of A soon with high probability” **A. Vergari**



$$p(\mathbf{X})$$

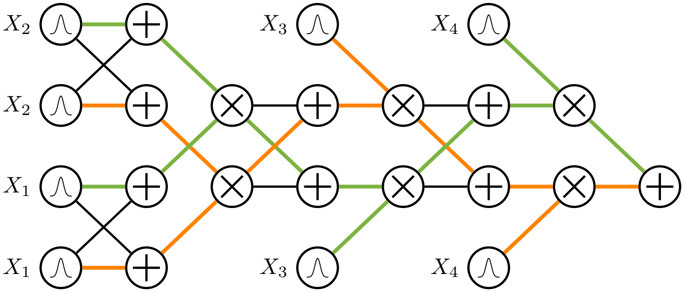


$$\sum_{i=1}^{K} w_i p_i(\mathbf{X})$$



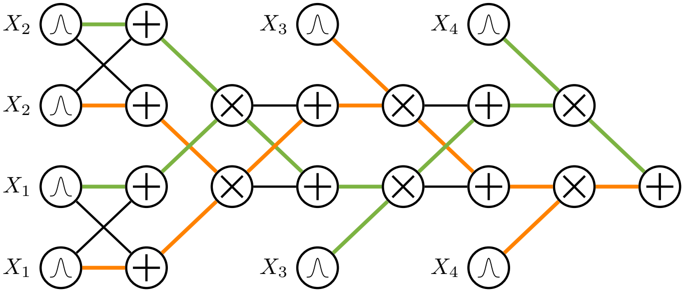
$$\sum_{i=1}^{2^D} w_i p_i(\mathbf{X}) = \mathsf{PC}(\mathbf{X})$$

expressive efficiency

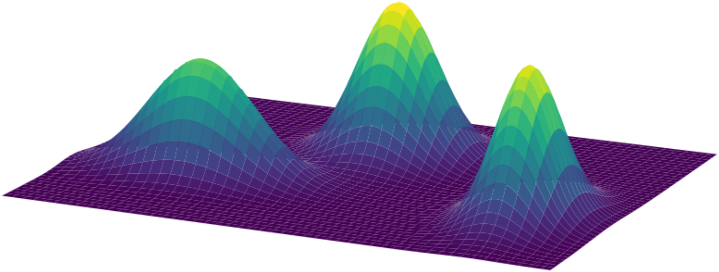


$$p(\mathbf{x}) = \sum_{\mathcal{T}} \left(\prod_{w_j \in \mathbf{w}_{\mathcal{T}}} w_j \right) \prod_{l \in \mathsf{leaves}(\mathcal{T})} p_l(\mathbf{x})$$

expressive efficiency

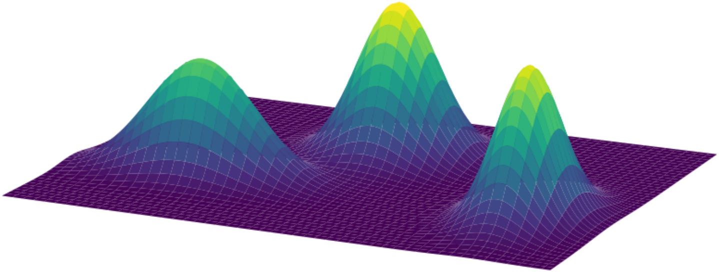


an exponential number of mixture components!



$$c(\mathbf{X}) = \sum_{i=1}^{K} w_i c_i(\mathbf{X}), \quad \text{with} \quad w_i \geq 0, \quad \sum_{i=1}^{K} w_i = 1$$

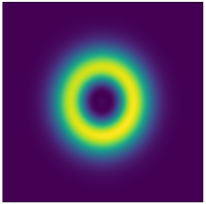
image taken from Hao Tang's course on ASR



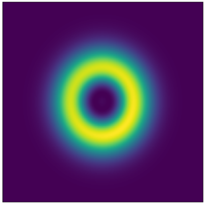
$$c(\mathbf{X}) = \sum_{i=1}^{K} w_i c_i(\mathbf{X}), \quad \text{with} \quad w_i \geq 0, \quad \sum_{i=1}^{K} w_i = 1$$

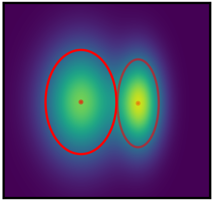
image taken from Hao Tang's course on ASR

however...



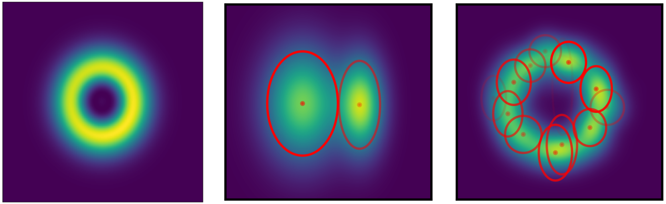
however...





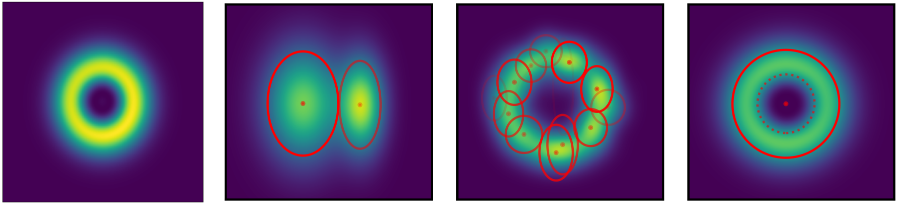
GMM ($K = 2$)

however...



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

however...



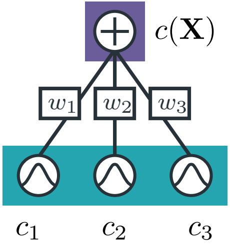
GMM ($K = 2$) GMM ($K = 16$) nGMM^2 ($K = 2$)

theorem

shallow mixtures
with negative parameters
can be *exponentially more compact* than
deep ones with positive ones.

Loconte et al., "Subtractive Mixture Models via Squaring: Representation and Learning", ICLR, 2024

subtractive MMs as circuits

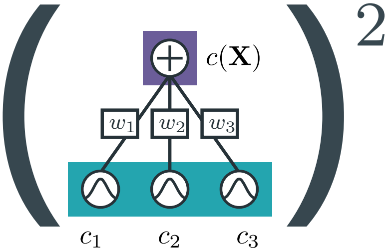


a **non-monotonic** smooth and (structured) decomposable circuit

$\Rightarrow$ *possibly with negative outputs*

$$c(\mathbf{X}) = \sum_{i=1}^{K} w_i c_i(\mathbf{X}), \qquad w_i \in \mathbb{R},$$

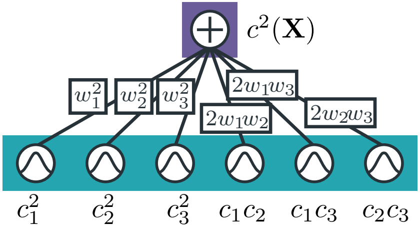
squaring shallow MMs



$$c^2(\mathbf{X}) = \left(\sum_{i=1}^{K} w_i c_i(\mathbf{X})\right)^2$$

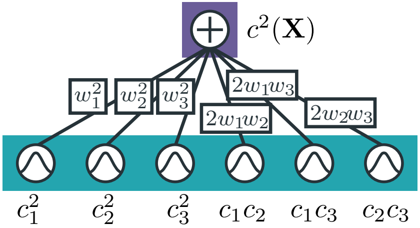
$\Longrightarrow$ *ensure non-negative output*

squaring shallow MMs



$$
\begin{aligned}
c^2(\mathbf{X}) &= \left(\sum_{i=1}^{K} w_i c_i(\mathbf{X})\right)^2 \\
&= \sum_{i=1}^{K} \sum_{j=1}^{K} w_i w_j c_i(\mathbf{X}) c_j(\mathbf{X})
\end{aligned}
$$

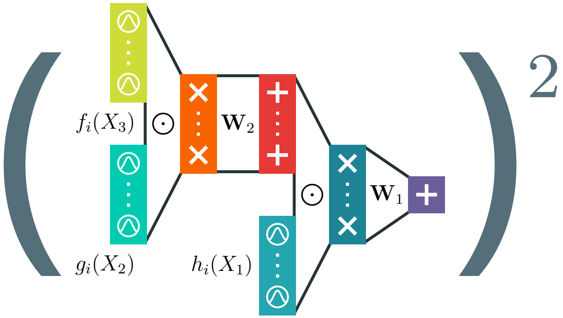
squaring shallow MMs



$$
\begin{aligned}
c^2(\mathbf{X}) &= \left(\sum_{i=1}^{K} w_i c_i(\mathbf{X})\right)^2 \\
&= \sum_{i=1}^{K} \sum_{j=1}^{K} w_i w_j c_i(\mathbf{X}) c_j(\mathbf{X})
\end{aligned}
$$

still a smooth and (str) decomposable PC with $\mathcal{O}(K^2)$ components!

$\Longrightarrow$ *but still* $\mathcal{O}(K)$ *parameters*

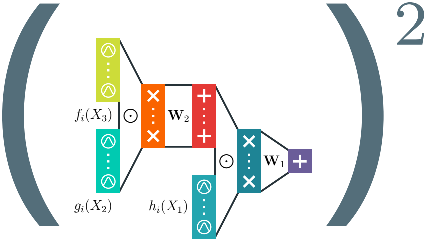


how to efficiently square (and renormalize) a deep PC?

Loconte et al., "Subtractive Mixture Models via Squaring: Representation and Learning", ICLR, 2024

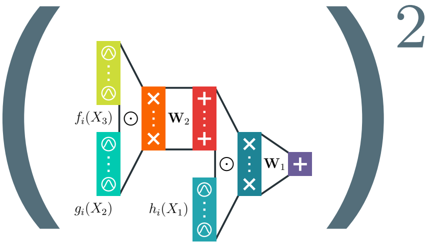
squaring deep PCs

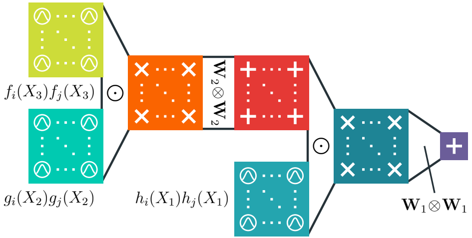
the tensorized way



squaring deep PCs

the tensorized way

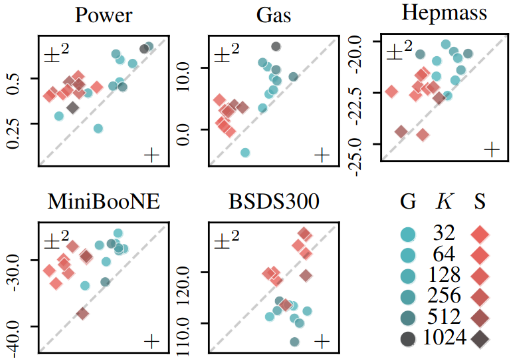


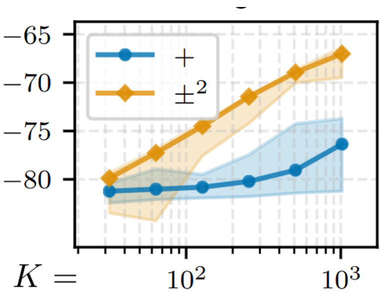


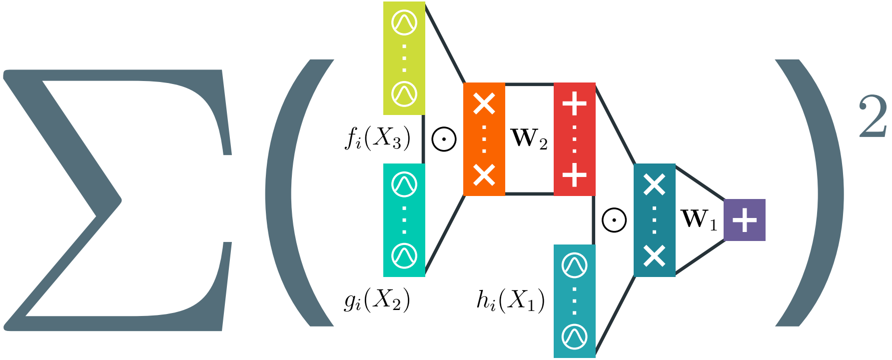
squaring a circuit = to squaring layers

how more expressive?

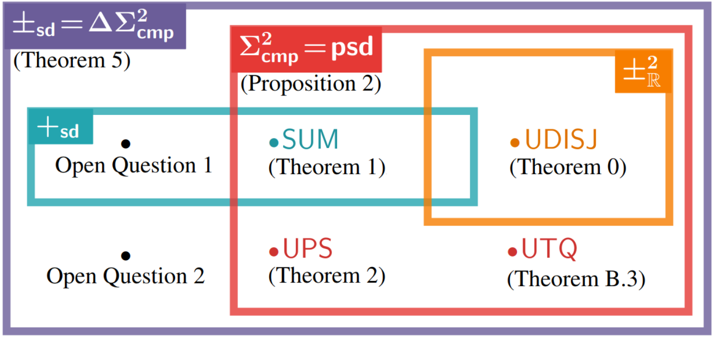
for the ML crowd



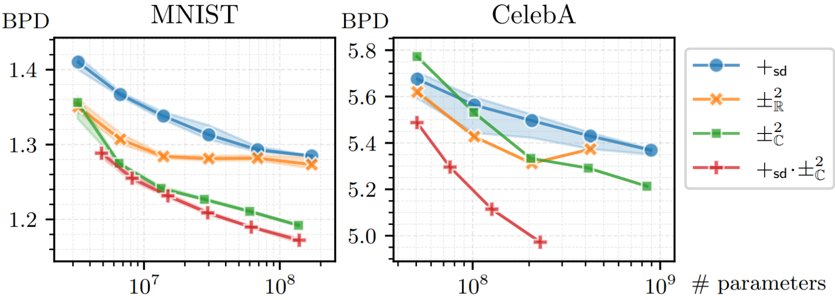




more that a single square?



SOS circuits are more expressive

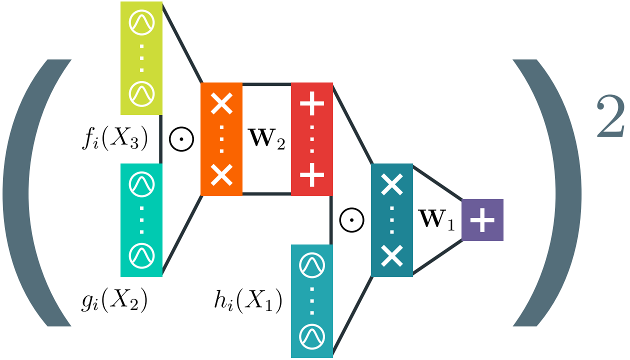


complex circuits are SOS (and scale better!)

Loconte, Mengel, and Vergari, "Sum of Squares Circuits", arXiv, 2024

cirkit

learning & reasoning with circuits in pytorch



questions?