UCLA Computer Science

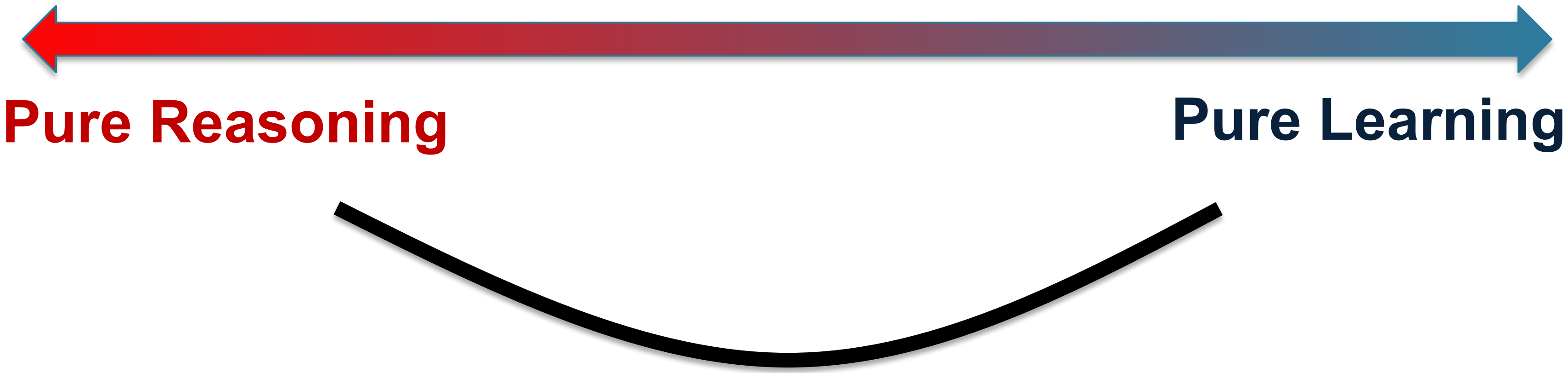
STAR AI
RESEARCH LAB
UCLA

AI can learn from data. But can it learn to reason?

Guy Van den Broeck

The 18th Reasoning Web Summer School - Sep 28 2022

The AI Dilemma



Integrate reasoning into modern deep learning algorithms

Outline

1. The paradox of learning to reason from data

   ~~deep learning~~

2. Tractable deep generative models

   probabilistic reasoning + *deep learning*

3. Learning with symbolic knowledge

   logical reasoning + *deep learning*

Outline

1. **The paradox of learning to reason from data**

   ~~deep learning~~

2. Tractable deep generative models

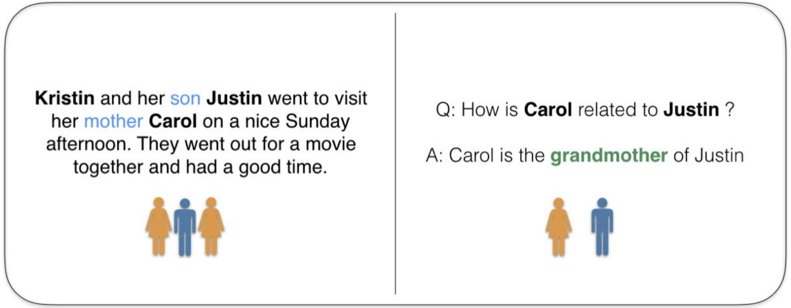
   probabilistic reasoning + *deep learning*

3. Learning with symbolic knowledge

   logical reasoning + *deep learning*

Can Language Models Perform Logical Reasoning?

Language Models achieve high performance on various “reasoning” benchmarks in NLP.



Reasoning Example from the CLUTRR dataset

It is unclear whether they solve the tasks following the rules of logical deduction.

<u>Language Models</u>:
input → ? → Carol is the grandmother of Justin.

<u>Reasoning</u>:
input → Justin in Kristin’s son; Carol is Kristin’s mother; → Carol is Justin’s mother’s mother; if X is Y’s mother’s mother then X is Y’s grandmother → Carol is the grandmother of Justin.

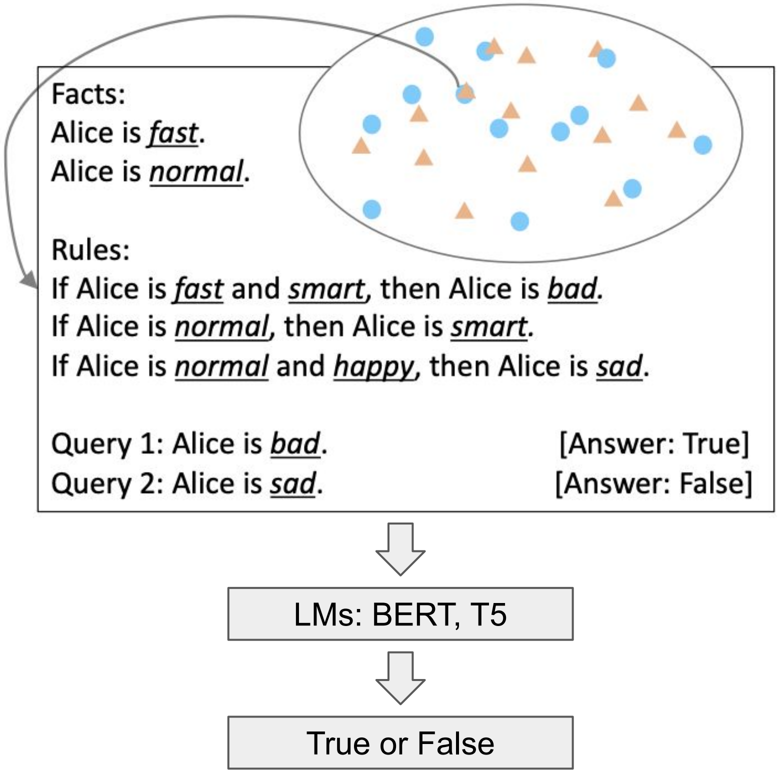
Problem Setting: SimpleLogic

Rules: If witty, then diplomatic. If careless and condemned and attractive, then blushing. If dishonest and inquisitive and average, then shy. If average, then stormy. If popular, then blushing. If talented, then hurt. If popular and attractive, then thoughtless. If blushing and shy and stormy, then inquisitive. If adorable, then popular. If cooperative and wrong and stormy, then thoughtless. If popular, then sensible. If cooperative, then wrong. If shy and cooperative, then witty. If polite and shy and thoughtless, then talented. If polite, then condemned. If polite and wrong, then inquisitive. If dishonest and inquisitive, then talented. If blushing and dishonest, then careless. If inquisitive and dishonest, then troubled. If blushing and stormy, then shy. If diplomatic and talented, then careless. If wrong and beautiful, then popular. If ugly and shy and beautiful, then stormy. If shy and inquisitive and attractive, then diplomatic. If witty and beautiful and frightened, then adorable. If diplomatic and cooperative, then sensible. If thoughtless and inquisitive, then diplomatic. If careless and dishonest and troubled, then cooperative. If hurt and witty and troubled, then dishonest. If scared and diplomatic and troubled, then average. If ugly and wrong and careless, then average. If dishonest and scared, then polite. If talented, then dishonest. If condemned, then wrong. If wrong and troubled and blushing, then scared. If attractive and condemned, then frightened. If hurt and condemned and shy, then witty. If cooperative, then attractive. If careless, then polite. If adorable and wrong and careless, then diplomatic. Facts: Alice sensible Alice condemned Alice thoughtless Alice polite Alice scared Alice average
Query: Alice is shy ?

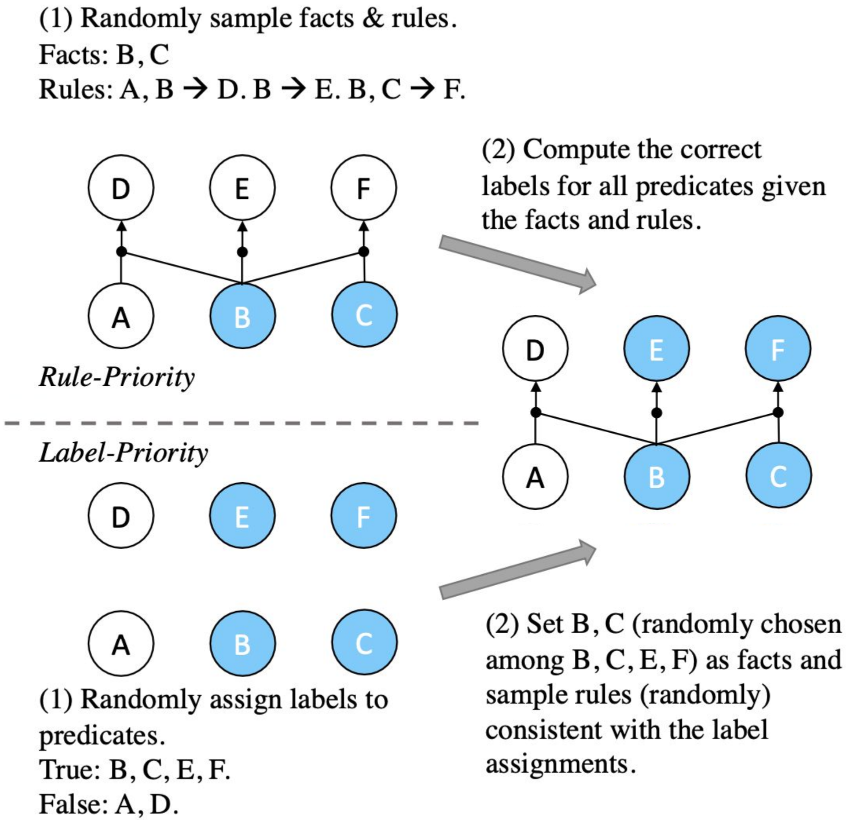
Problem Setting: SimpleLogic

The easiest of reasoning problems:

1. **Propositional logic** fragment
   a. bounded vocabulary & number of rules
   b. bounded reasoning depth (≤ 6)
   c. finite space (≈ 10^360)
2. **No language variance**: templated language
3. **Self-contained**
   No prior knowledge
4. **Purely symbolic** predicates
   No shortcuts from word meaning
5. **Tractable** logic (definite clauses)
   Can always be solved efficiently



Training a BERT model on SimpleLogic



Test accuracy for different reasoning depths

Test	0	1	2	3	4	5	6
RP	99.9	99.8	99.7	99.3	98.3	97.5	95.5

Test	0	1	2	3	4	5	6
LP	100.0	100.0	99.9	99.9	99.7	99.7	99.0

Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang and Guy Van den Broeck. On the Paradox of Learning to Reason from Data, 2022

Has BERT learned to reason from data?

1. Easiest of reasoning problems (no variance, self-contained, purely symbolic, tractable)
2. RP/LP data covers the whole problem space
3. The learned model has almost 100% test accuracy
4. There exist BERT parameters that compute the ground-truth reasoning function:

   <u>Theorem 1:</u> *For a BERT model with* n *layers and 12 attention heads, by construction, there exists a set of parameters such that the model can correctly solve any reasoning problem in SimpleLogic that requires at most* $n - 2$ *steps of reasoning.*

Surely, under these conditions,
BERT has learned the ground-truth reasoning function!

Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang and Guy Van den Broeck. On the Paradox of Learning to Reason from Data, 2022

The Paradox of Learning to Reason from Data

Train	Test	0	1	2	3	4	5	6
RP	RP	99.9	99.8	99.7	99.3	98.3	97.5	95.5
	LP	99.8	99.8	99.3	96.0	90.4	75.0	57.3
LP	RP	97.3	66.9	53.0	54.2	59.5	65.6	69.2
	LP	100.0	100.0	99.9	99.9	99.7	99.7	99.0

The BERT model trained on one distribution fails to generalize to the other distribution within the same problem space.

1. If BERT **has learned** to reason,
   it should not exhibit such generalization failure.

2. If BERT **has not learned** to reason,
   it is baffling how it achieves near-perfect in-distribution test accuracy.

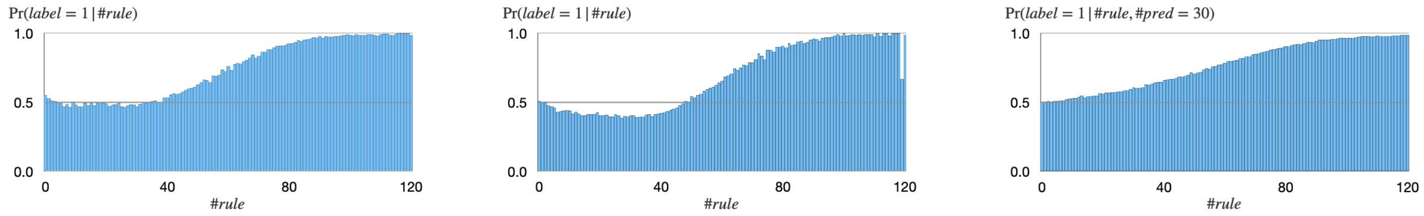
Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang and Guy Van den Broeck. On the Paradox of Learning to Reason from Data, 2022

Why? Statistical Features

Monotonicity of entailment:
Any rules can be freely added to the hypothesis of any proven fact.

The more rules given, the more likely a predicate will be proved.

Pr(label = True | Rule # = x) should increase (roughly) monotonically with x



(a) Statistics for examples generated by Rule-Priority (RP). (b) Statistics for examples generated by Label-Priority (LP). (c) Statistics for examples generated by uniform sampling;

BERT leverages statistical features to make predictions

RP_b downsamples from RP such that Pr(label = True | rule# = x) = 0.5 for all x

Train	Test	0	1	2	3	4	5	6
RP	RP	99.9	99.8	99.7	99.3	98.3	97.5	95.5
	RP_b	99.0	99.3	98.5	97.5	96.7	93.5	88.3

1. Accuracy drop from RP to RP_b indicates that
   the model is using rule# as a statistical feature to make predictions.

2. Though removing one statistical feature from training data can help with model generalization, there are potentially countless statistical features and it is computationally infeasible to jointly remove them.

Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang and Guy Van den Broeck. On the Paradox of Learning to Reason from Data, 2022

First Conclusion

Experiments unveil the fundamental difference between

1. learning to reason, and
2. learning to achieve high performance on benchmarks using statistical features.

Be careful deploying AI in applications where this difference matters.

Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang and Guy Van den Broeck. On the Paradox of Learning to Reason from Data, 2022

Outline

1. The paradox of learning to reason from data

   ~~deep learning~~

2. **Tractable deep generative models**

   probabilistic reasoning + *deep learning*

3. Learning with symbolic knowledge

   logical reasoning + *deep learning*

Fully factorized

NaiveBayes AndOrGraphs PDGs

Trees PSDDs CNets LTMs SPNs NADEs

Thin Junction Trees ACs MADEs MAFs VAEs

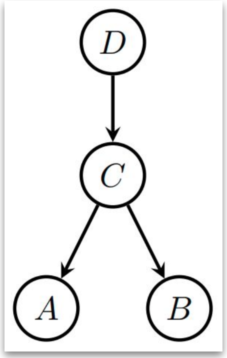
DPPs FVSBNs TACs IAFs NAFs RAEs

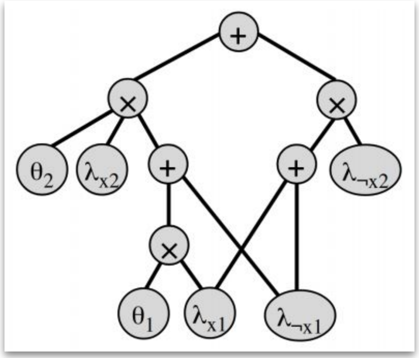
Mixtures BNs NICE FGs GANs

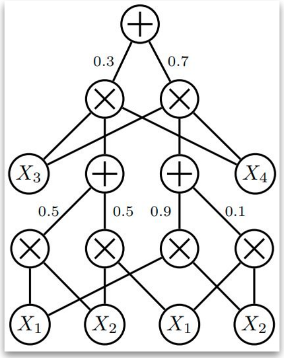
RealNVP MNs

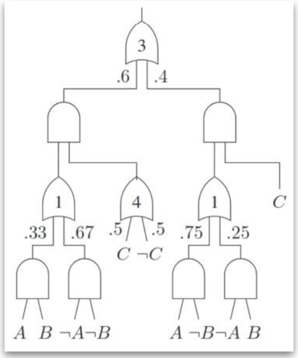
Intractable and *tractable* models

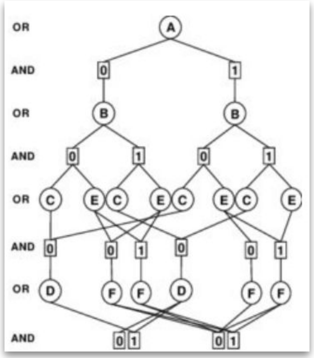
Tractable Probabilistic Models

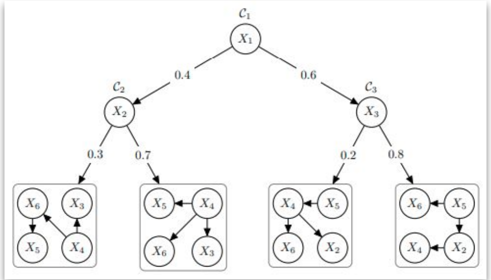


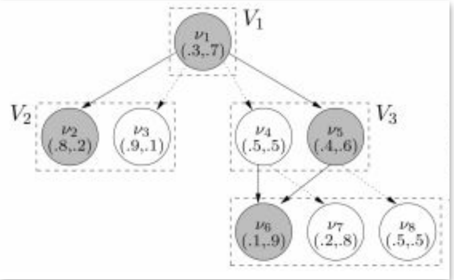


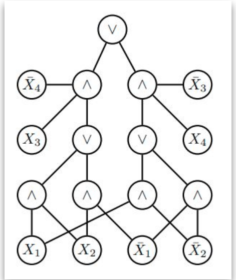


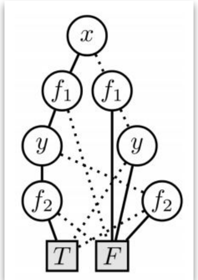


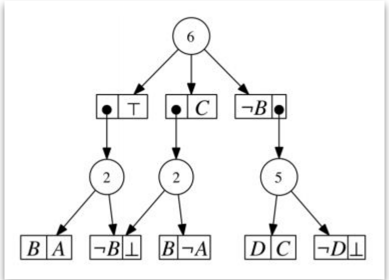












"Every talk needs a joke and a literature overview slide, not necessarily distinct"

\- after Ron Graham

Probabilistic circuits

Fully factorized

NaiveBayes AndOrGraphs PDGs

Trees PSDDs CNets LTMs SPNs NADEs

Thin Junction Trees ACs MADEs MAFs VAEs

DPPs FVSBNs TACs IAFs NAFs RAEs

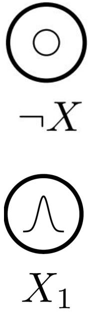
Mixtures BNs NICE FGs GANs

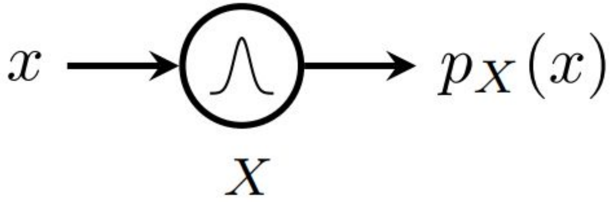
RealNVP MNs

a *unifying framework* for tractable models

Probabilistic circuits

computational graphs that recursively define distributions



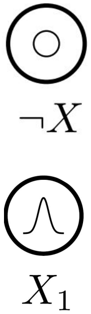


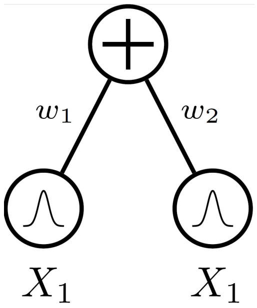
Simple distributions are tractable “black boxes” for:

- EVI: output $p(\mathbf{x})$ (density or mass)
- MAR: output 1 (normalized) or Z (unnormalized)
- MAP: output the mode

Probabilistic circuits

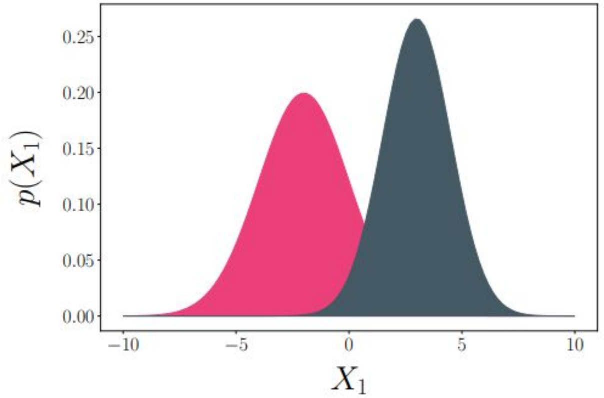
computational graphs that recursively define distributions





$$p(X_1) = w_1 p_1(X_1) + w_2 p_2(X_1)$$

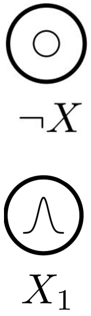
⇒ *mixtures*

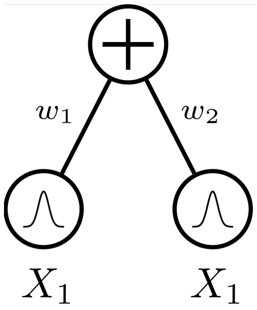


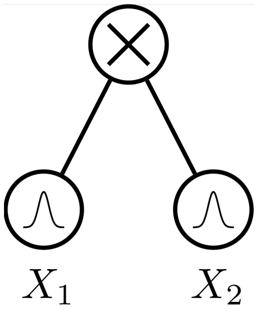
$$p(X) = p(Z = \boxed{1}) \cdot p_1(X|Z = \boxed{1}) + p(Z = \boxed{2}) \cdot p_2(X|Z = \boxed{2})$$

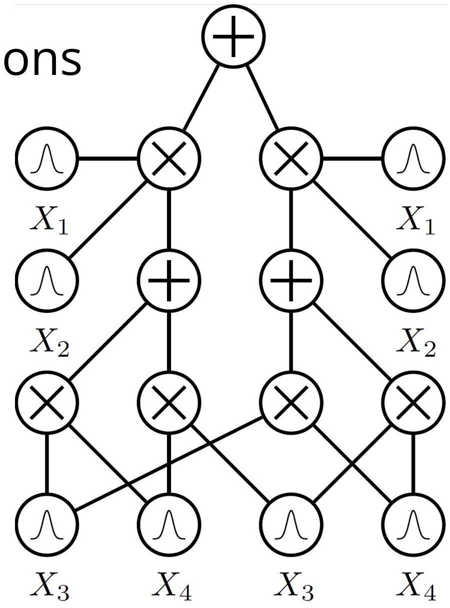
Probabilistic circuits

computational graphs that recursively define distributions









$$p(X_1) = w_1 p_1(X_1) + w_2 p_2(X_1)$$

$\Rightarrow$ *mixtures*

$$p(X_1, X_2) = p(X_1) \cdot p(X_2)$$

$\Rightarrow$ *factorizations*

Tractable Probabilistic Inference

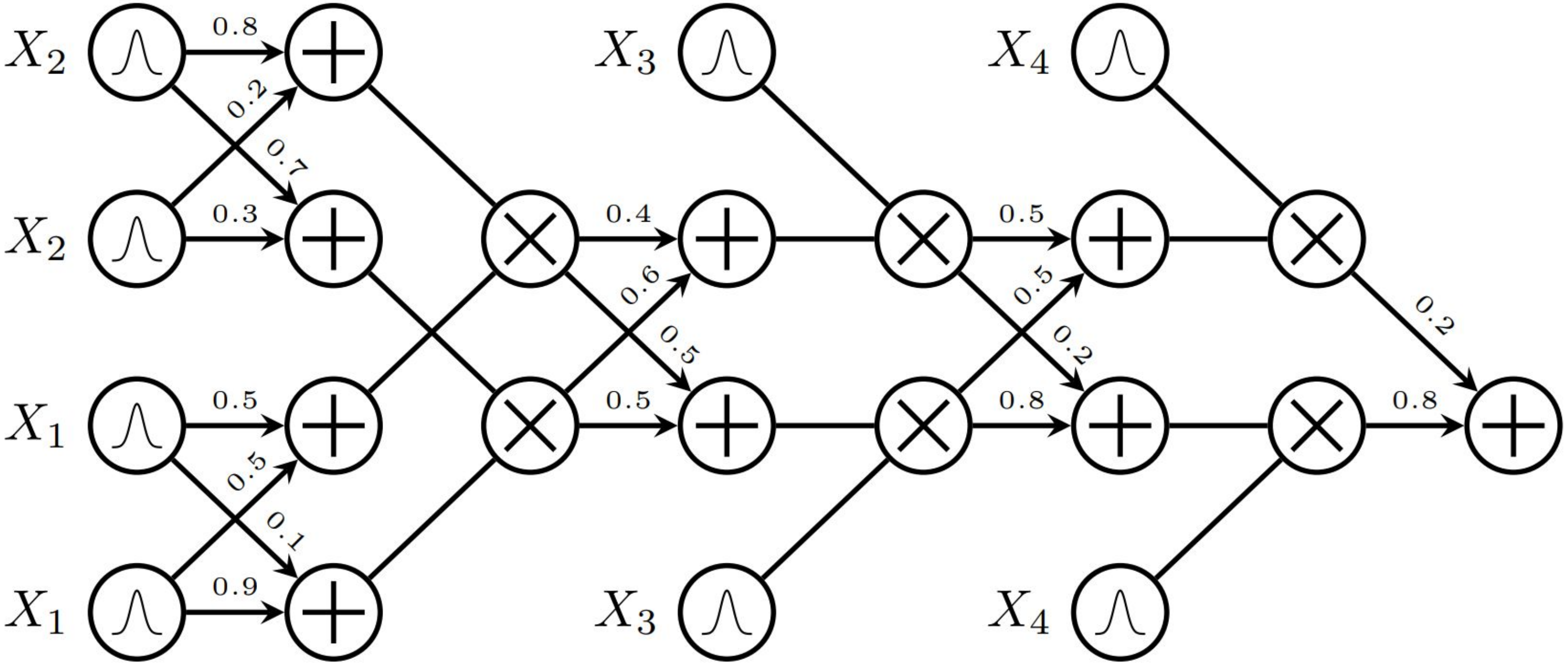
A class of queries $\mathcal{Q}$ *is* tractable *on a family of probabilistic models* $\mathcal{M}$ *iff for any query* $\mathbf{q} \in \mathcal{Q}$ *and model* $\mathbf{m} \in \mathcal{M}$ ***exactly*** *computing* $\mathbf{q}(\mathbf{m})$ *runs in time* $O(\text{poly}(|\mathbf{m}|))$.

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

$\Rightarrow$ *Note: if* $\mathcal{M}$ *is compact in the number of random variables* $\mathbf{X}$*, that is,* $|\mathbf{m}| \in O(\text{poly}(|\mathbf{X}|))$*, then query time is* $O(\text{poly}(|\mathbf{X}|))$.

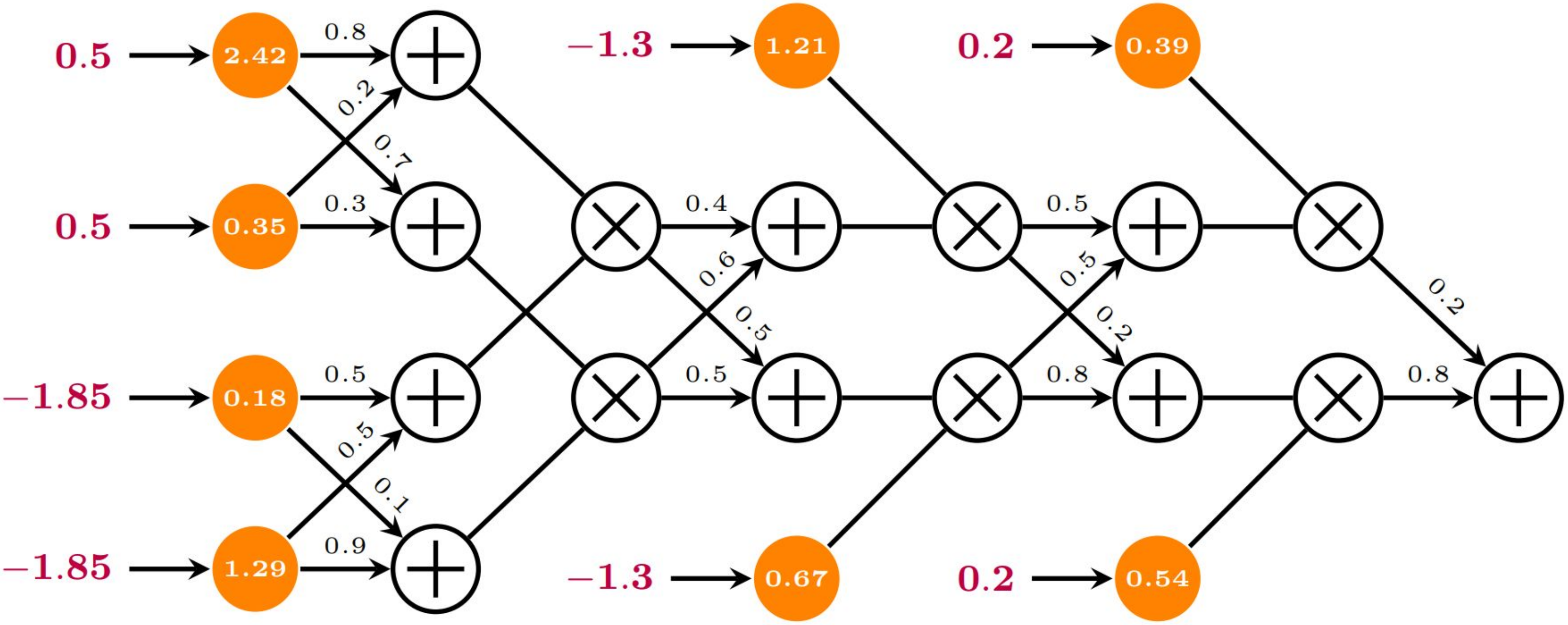
Likelihood

$$p(X_1 = \mathbf{-1.85}, X_2 = \mathbf{0.5}, X_3 = \mathbf{-1.3}, X_4 = \mathbf{0.2})$$



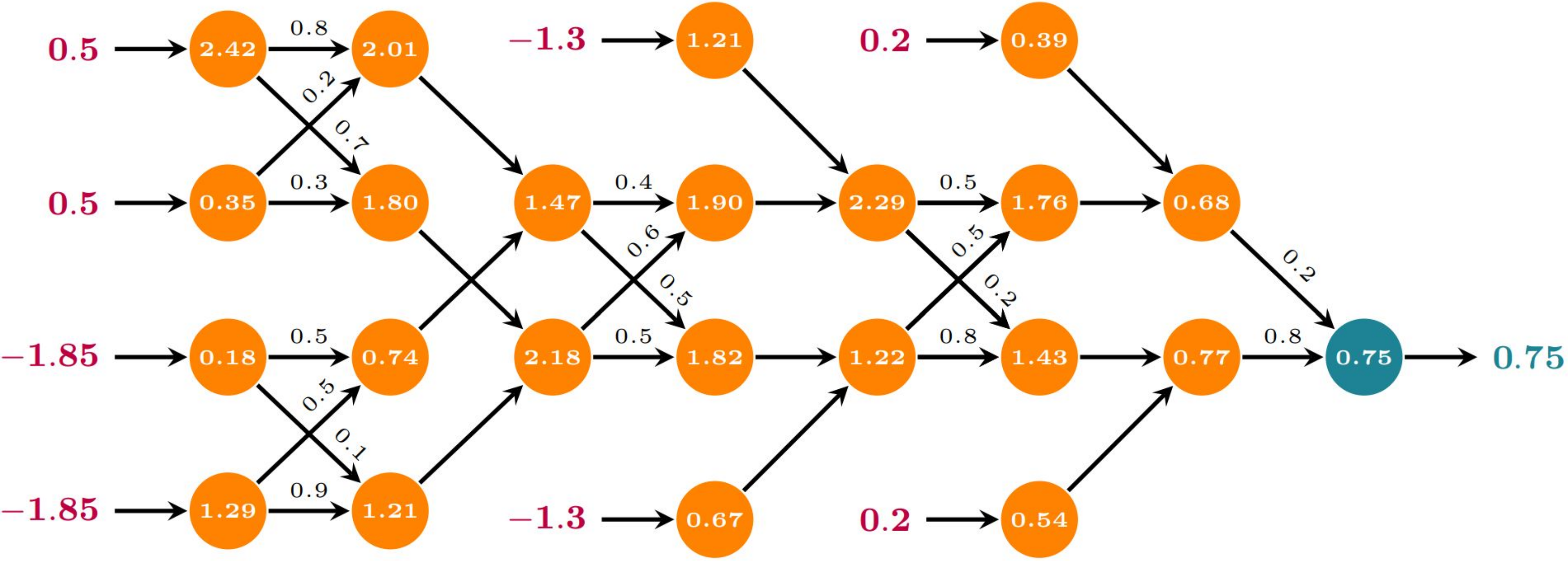
Likelihood

$p(X_1 = \mathbf{-1.85}, X_2 = \mathbf{0.5}, X_3 = \mathbf{-1.3}, X_4 = \mathbf{0.2})$

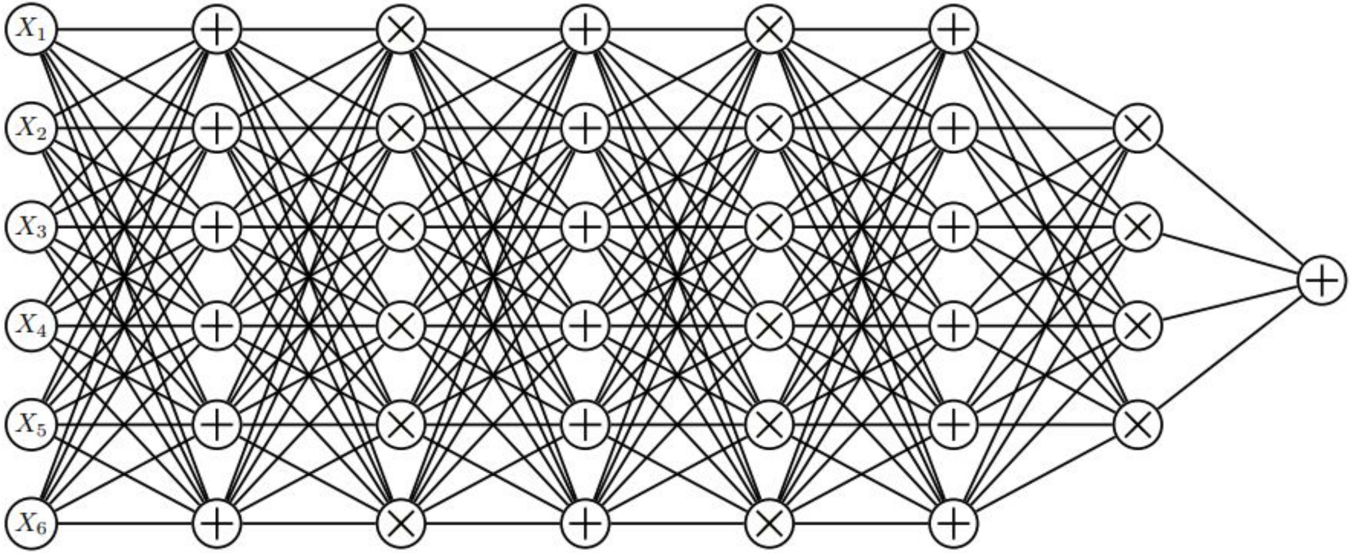


Likelihood

$$p(X_1 = \mathbf{-1.85}, X_2 = \mathbf{0.5}, X_3 = \mathbf{-1.3}, X_4 = \mathbf{0.2})$$

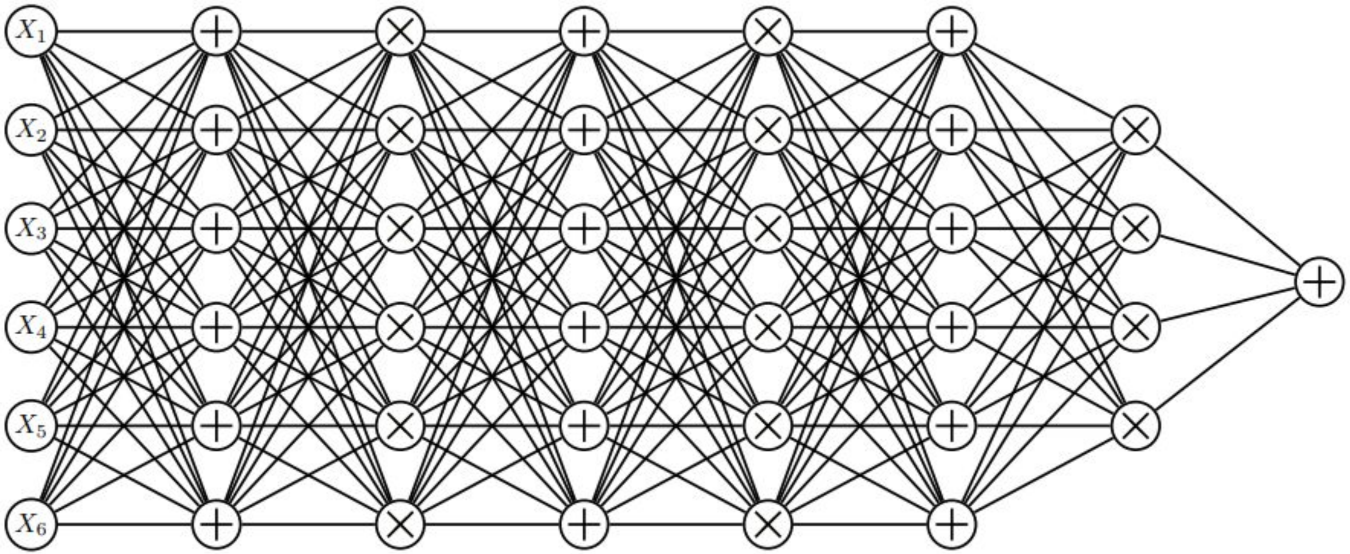


Just sum, products and distributions?



just arbitrarily compose them like a neural network!

Just sum, products and distributions?



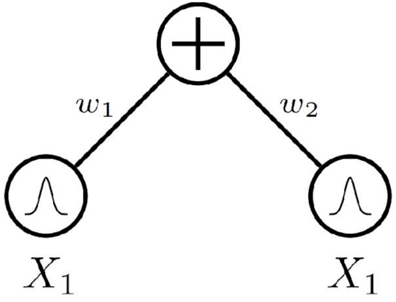
~~*just arbitrarily compose them like a neural network!*~~

$\Rightarrow$ structural constraints needed for tractability

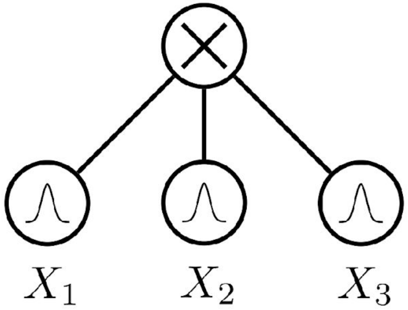
Tractable marginals

A sum node is ***smooth*** if its children depend on the same set of variables.

A product node is ***decomposable*** if its children depend on disjoint sets of variables.



smooth circuit



decomposable circuit

Darwiche and Marquis, “A Knowledge Compilation Map”, 2002

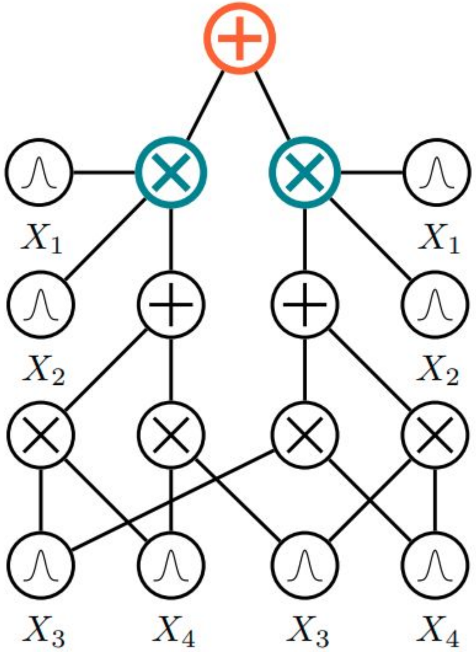
Smoothness + decomposability = tractable MAR

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

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

$$= \sum_i w_i \int p_i(\mathbf{x}) d\mathbf{x}$$

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



[Darwiche & Marquis JAIR 2001, Poon & Domingos UAI11]

Smoothness + **decomposability** = **tractable MAR**

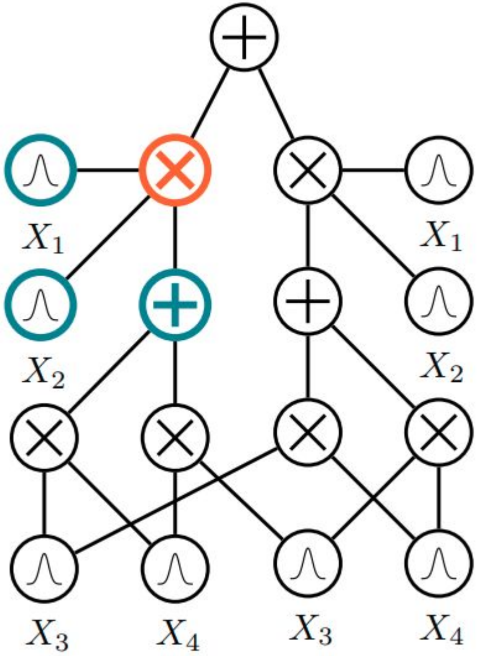
If $p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{x})p(\mathbf{y})p(\mathbf{z})$, (***decomposability***):

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

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

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

$\Rightarrow$ *integrals decompose into easier ones*



Smoothness + decomposability = tractable MAR

Forward pass evaluation for MAR

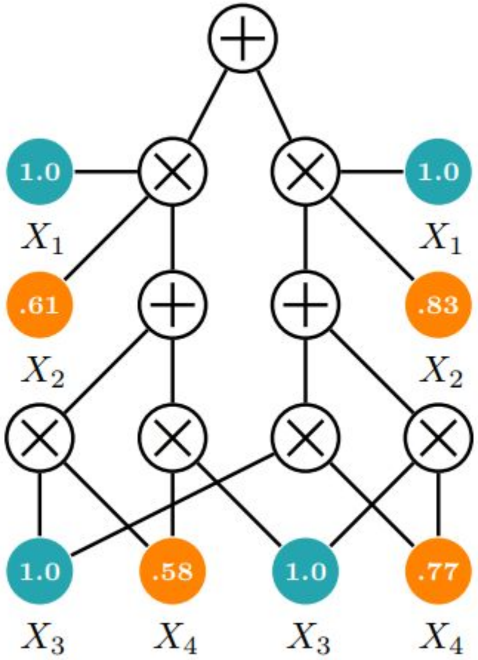
$\Rightarrow$ *linear in circuit size!*

E.g. to compute $p(x_2, x_4)$:

- leafs over X_1 and X_3 output $Z_i = \int p(x_i)dx_i$

  $\Rightarrow$ *for normalized leaf distributions:* 1.0

- leafs over X_2 and X_4 output ***EVI***
- feedforward evaluation (bottom-up)



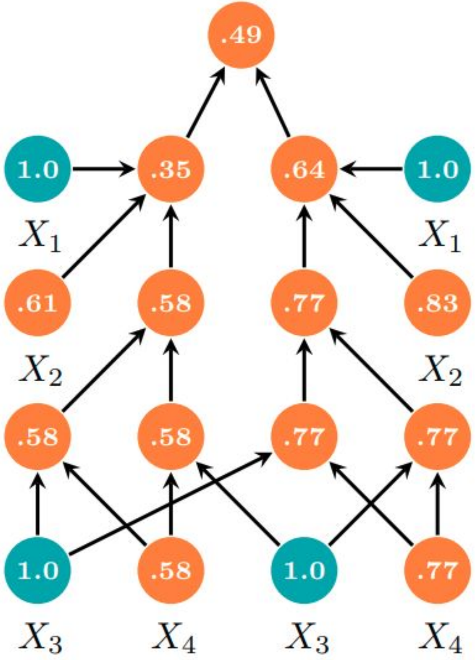
Smoothness + decomposability = tractable MAR

Forward pass evaluation for MAR

$\Rightarrow$ *linear in circuit size!*

E.g. to compute $p(x_2, x_4)$:

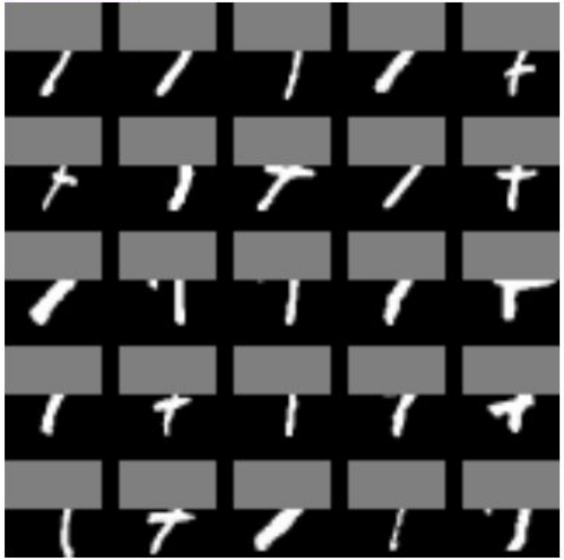
- leafs over X_1 and X_3 output $Z_i = \int p(x_i)dx_i$
  $\Rightarrow$ *for normalized leaf distributions:* 1.0
- leafs over X_2 and X_4 output ***EVI***
- feedforward evaluation (bottom-up)



Tractable MAR on PCs (Einsum Networks)

EVI $10,958.72\,\text{nats}$

MAR $5,387.55\,\text{nats}$



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

Smoothness + **decomposability** = ~~**tractable MAP**~~

We ***cannot*** decompose bottom-up a MAP query:

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

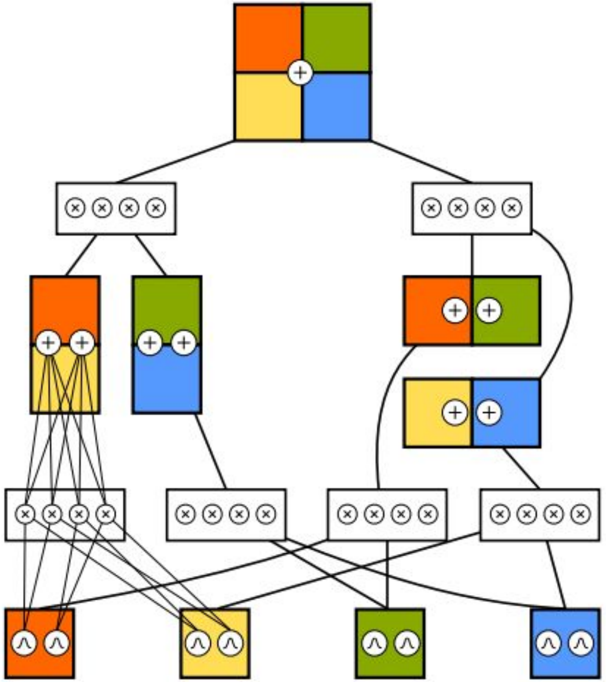
since for a sum node we are marginalizing out a latent variable

$$\max_{\mathbf{q}} \sum_{i} w_i p_i(\mathbf{q}, \mathbf{e}) = \max_{\mathbf{q}} \sum_{\mathbf{z}} p(\mathbf{q}, \mathbf{z}, \mathbf{e}) \neq \sum_{\mathbf{z}} \max_{\mathbf{q}} p(\mathbf{q}, \mathbf{z}, \mathbf{e})$$

$\Longrightarrow$ *MAP for latent variable models is* **intractable** *[Conaty et al. 2017]*

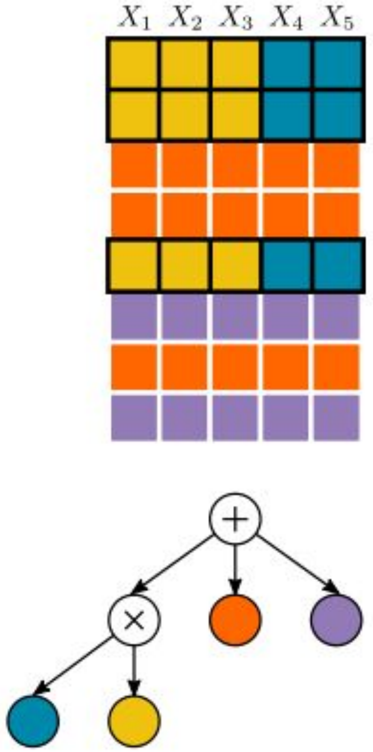
Where do architectures come from?

Where do architectures come from?



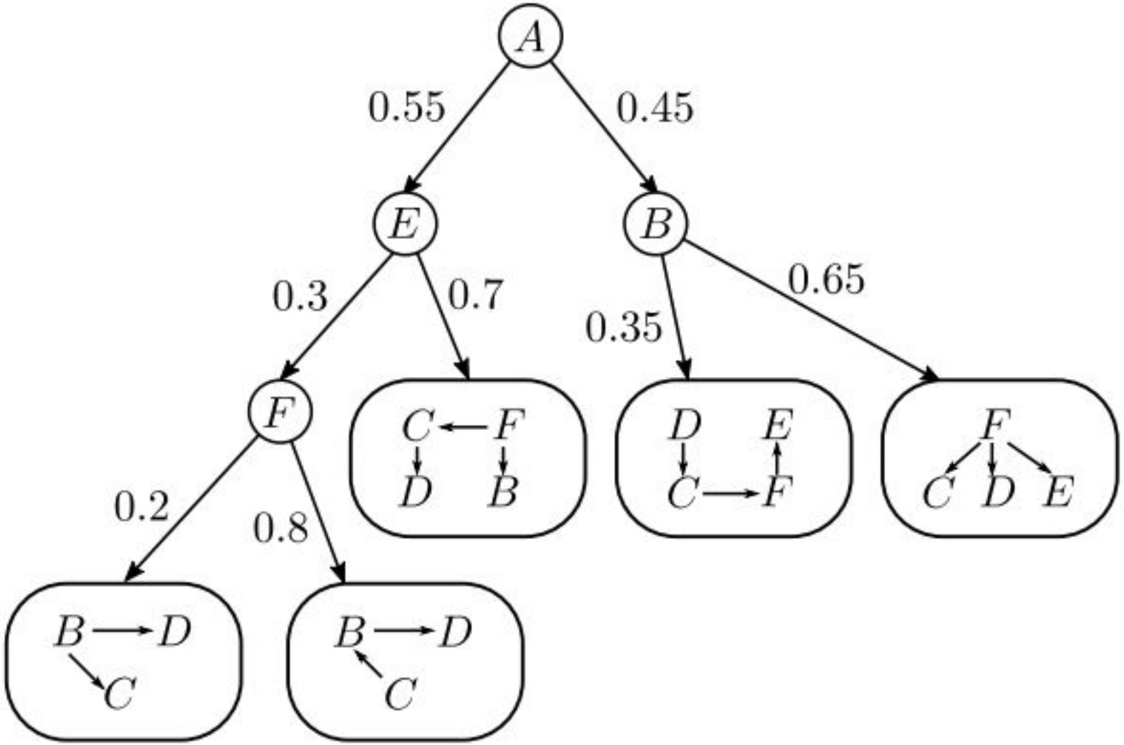
Poon and Domingos, "Sum-Product Networks: a New Deep Architecture", 2011

Where do architectures come from?



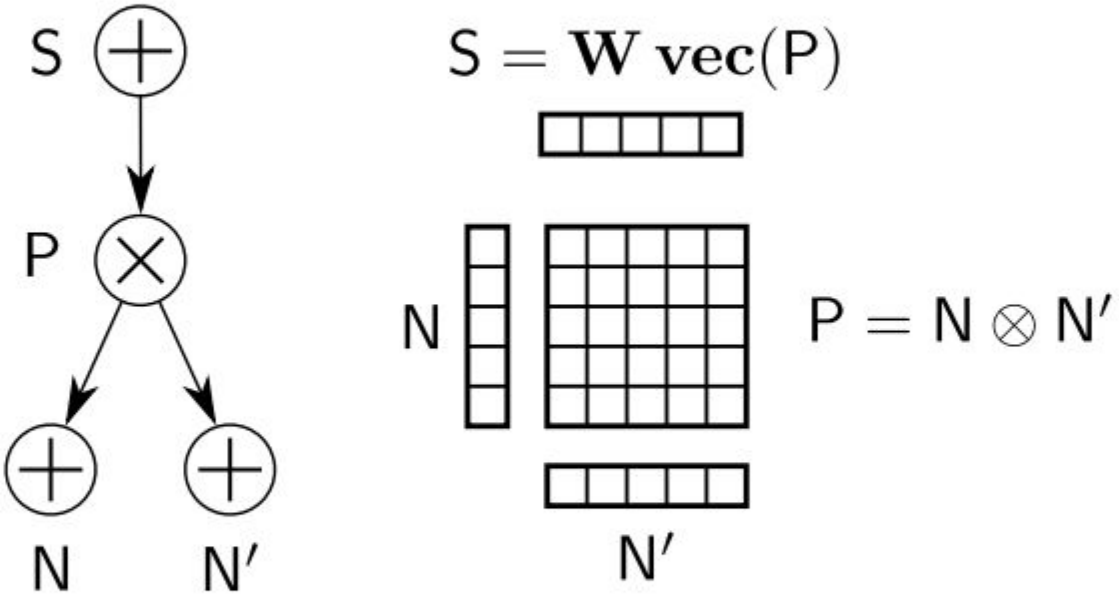
Gens and Domingos, "Learning the Structure of Sum-Product Networks", 2013

Where do architectures come from?



[Rahman et al. 2014]

Where do architectures come from?

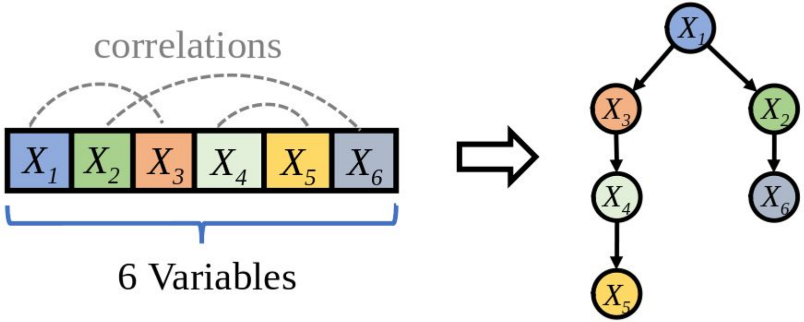


Einsum Networks: Fast and Scalable Learning of Tractable Probabilistic Circuits

Learning Expressive Probabilistic Circuits

Hidden Chow-Liu Trees

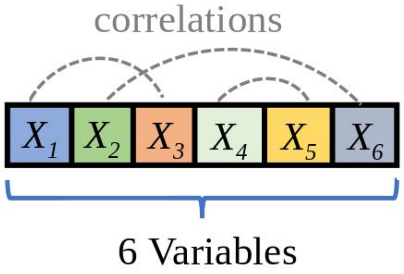
Learned **CLT structure** captures strong pairwise dependencies



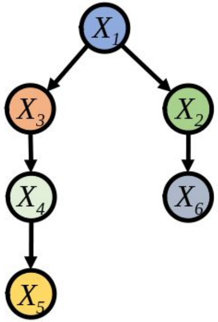
Anji Liu and Guy Van den Broeck. Tractable Regularization of Probabilistic Circuits, *NeurIPS*, 2021.

Learning Expressive Probabilistic Circuits

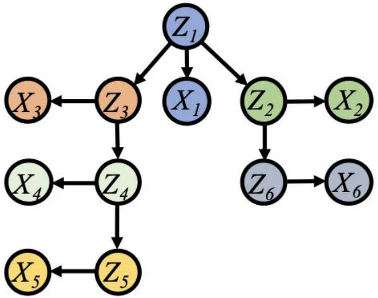
Hidden Chow-Liu Trees



Learned **CLT structure** captures strong pairwise dependencies



Learned **HCLT structure**

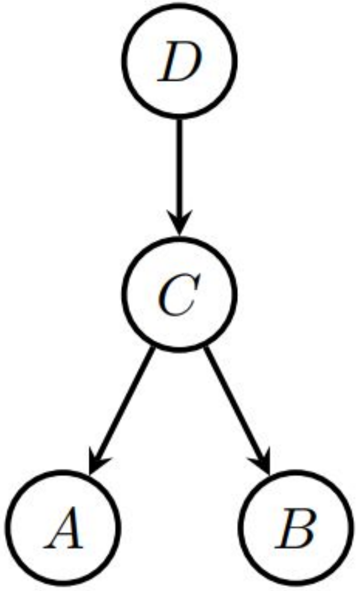


Compile into an equivalent PC

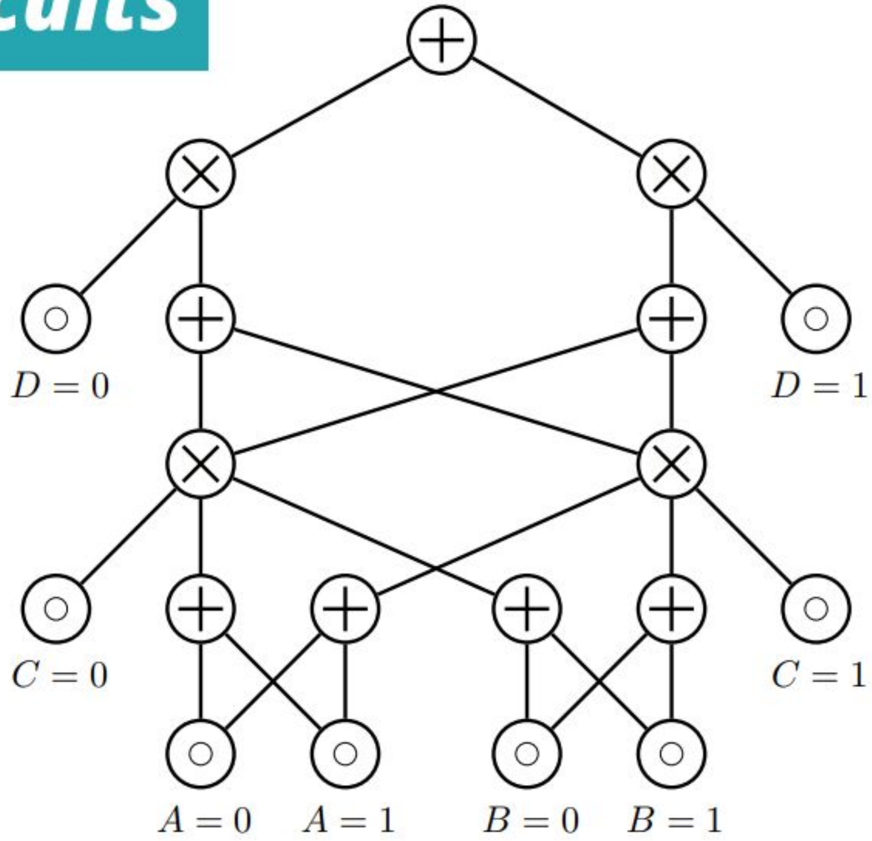
Anji Liu and Guy Van den Broeck. Tractable Regularization of Probabilistic Circuits, *NeurIPS*, 2021.

From BN trees to circuits

via compilation

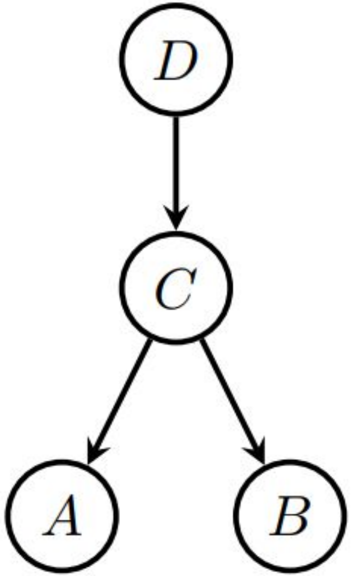


$\rightarrow$

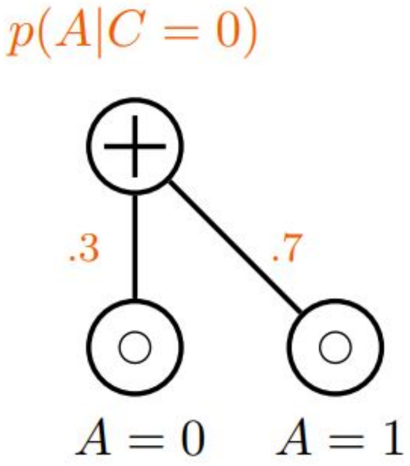


From BN trees to circuits

via compilation

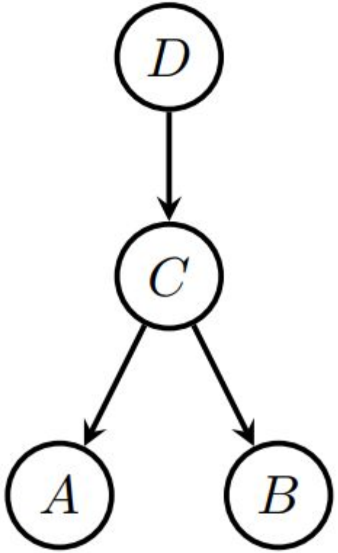


...compile a leaf CPT

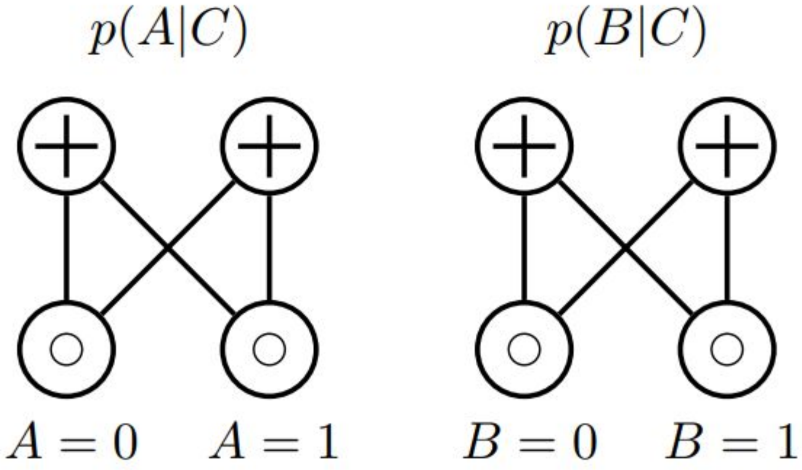


From BN trees to circuits

via compilation

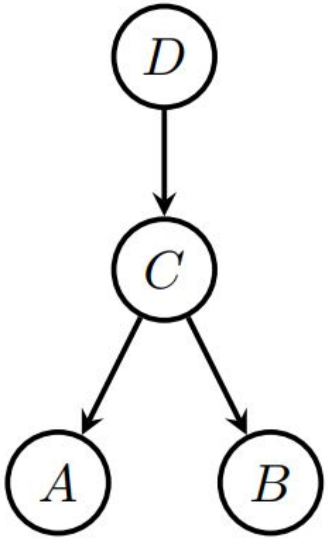


...compile a leaf CPT...for all leaves...

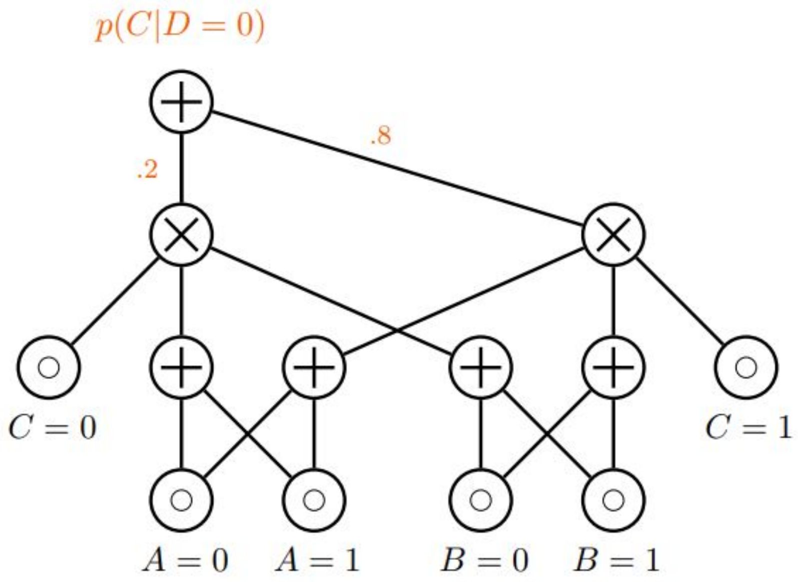


From BN trees to circuits

via compilation

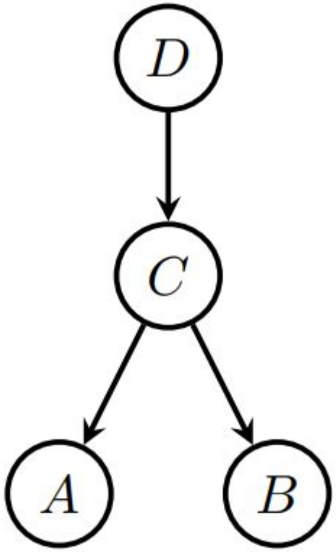


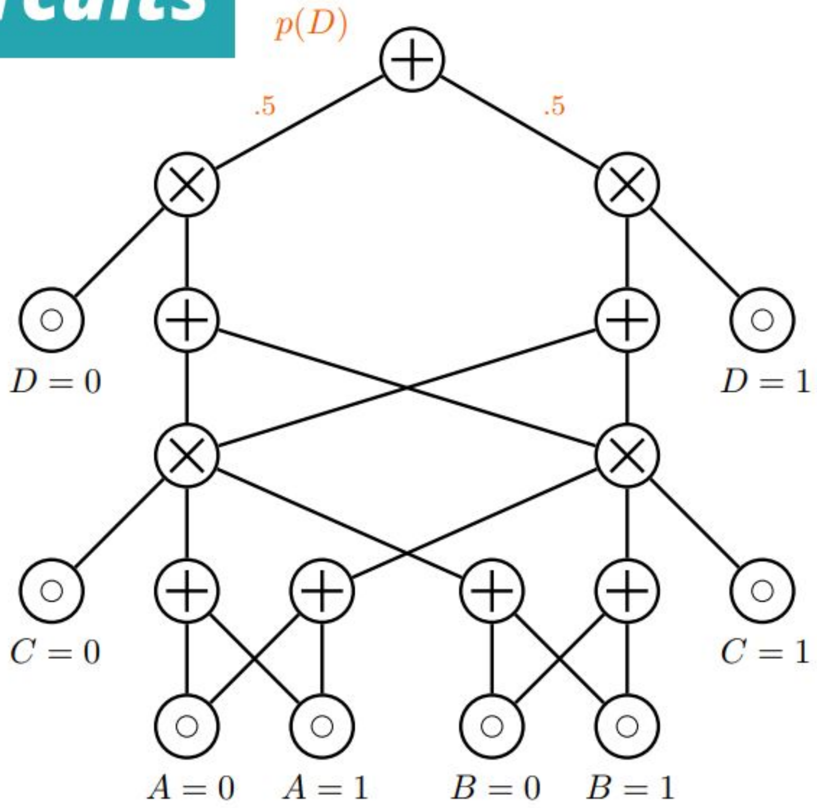
...and recurse over parents...



From BN trees to circuits

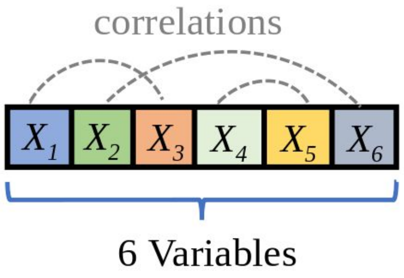
via compilation



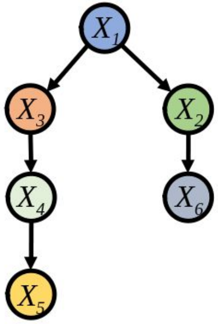


Learning Expressive Probabilistic Circuits

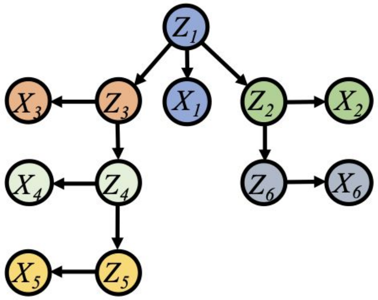
Hidden Chow-Liu Trees



Learned **CLT structure** captures strong pairwise dependencies



Learned **HCLT structure**

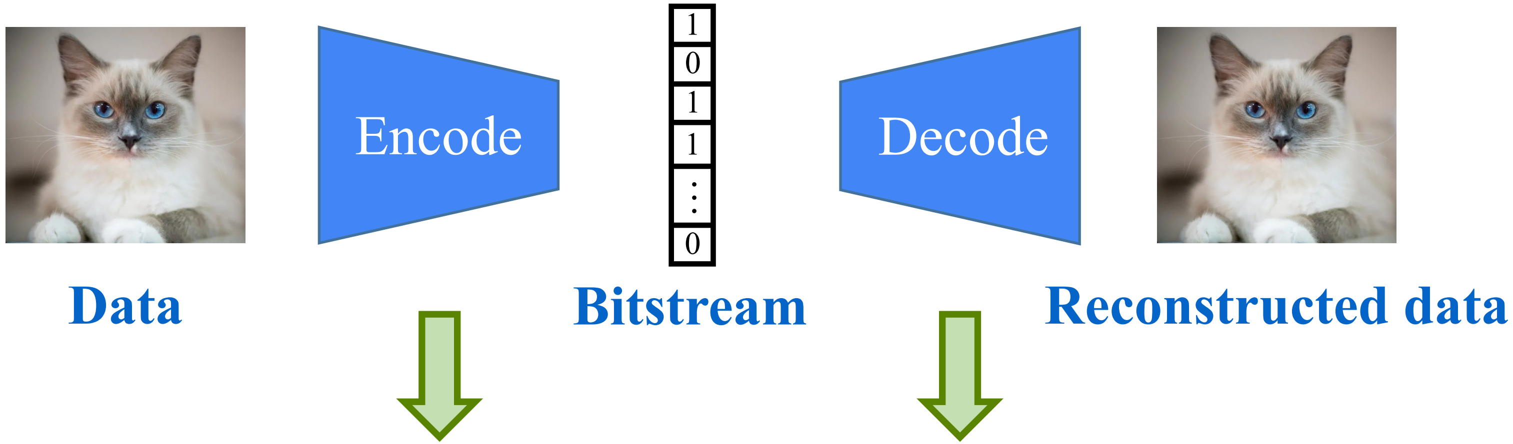


Compile into an equivalent PC

Mini-batch Stochastic **Expectation Maximization**

Anji Liu and Guy Van den Broeck. Tractable Regularization of Probabilistic Circuits, *NeurIPS*, 2021.

Lossless Data Compression



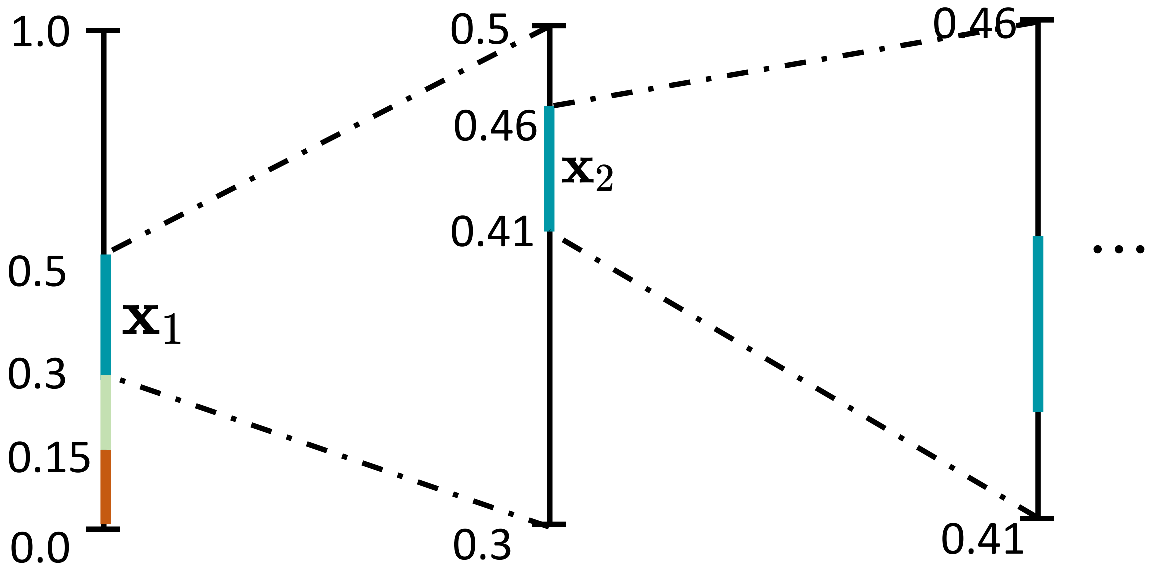
Expressive probabilistic model $p(\boldsymbol{x})$ ⇨ Determines the theoretical limit of compression rate

+

Efficient coding algorithm ⇨ How close we can approach the theoretical limit

A Typical Streaming Code – Arithmetic Coding

We want to compress a set of variables (e.g., pixels, letters) $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$



Compress $\mathbf{x}_1$ with $-\log p(\mathbf{x}_1)$ bits

Compress $\mathbf{x}_2$ with $-\log p(\mathbf{x}_2|\mathbf{x}_1)$ bits

Compress $\mathbf{x}_3$ with $-\log p(\mathbf{x}_3|\mathbf{x}_1, \mathbf{x}_2)$ bits

Need to compute

$p(X_1 < x_1)$

$p(X_1 \leq x_1)$

$p(X_2 < x_2|x_1)$

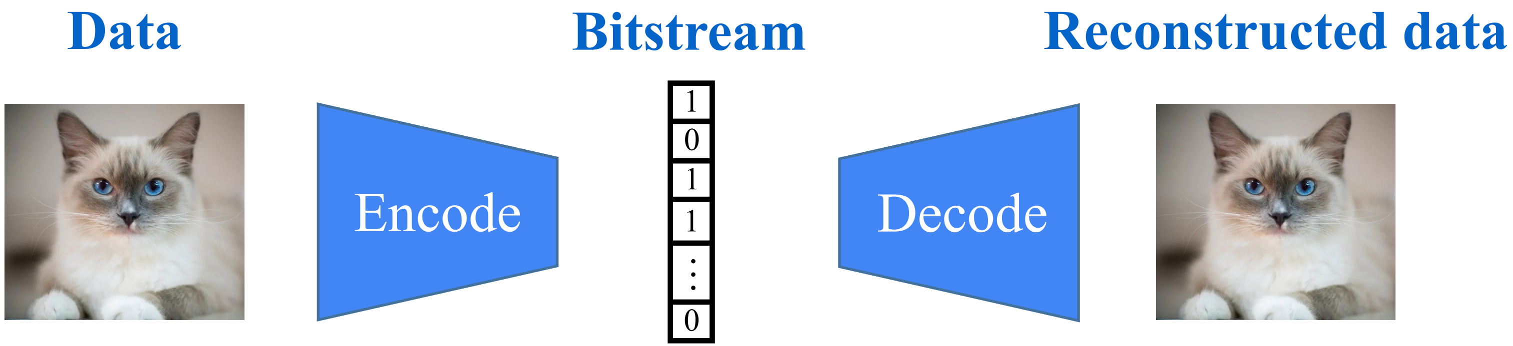
$p(X_2 \leq x_2|x_1)$

$p(X_3 < x_3|x_1, x_2)$

$p(X_3 \leq x_3|x_1, x_2)$

$\vdots$

Lossless Neural Compression with Probabilistic Circuits



Probabilistic Circuits

- Expressive → SoTA likelihood on MNIST.
- Fast → Time complexity of en/decoding is **O(|p| log(D))**, where D is the # variables and |p| is the size of the PC.

Arithmetic Coding:

$$p(X_1 < x_1)$$
$$p(X_1 \leq x_1)$$
$$p(X_2 < x_2 | x_1)$$
$$p(X_2 \leq x_2 | x_1)$$
$$p(X_3 < x_3 | x_1, x_2)$$
$$p(X_3 \leq x_3 | x_1, x_2)$$
$$\vdots$$

Anji Liu, Stephan Mandt and Guy Van den Broeck. Lossless Compression with Probabilistic Circuits, 2021.

Lossless Neural Compression with Probabilistic Circuits

SoTA compression rates

Dataset	HCLT (ours)	IDF	BitSwap	BB-ANS	JPEG2000	WebP	McBits
MNIST	**1.24** (1.20)	1.96 (1.90)	1.31 (1.27)	1.42 (1.39)	3.37	2.09	(1.98)
FashionMNIST	3.37 (3.34)	3.50 (3.47)	**3.35** (3.28)	3.69 (3.66)	3.93	4.62	(3.72)
EMNIST (Letter)	**1.84** (1.80)	2.02 (1.95)	1.90 (1.84)	2.29 (2.26)	3.62	3.31	(3.12)
EMNIST (ByClass)	**1.89** (1.85)	2.04 (1.98)	1.91 (1.87)	2.24 (2.23)	3.61	3.34	(3.14)

Compress and decompress 5-40x faster than NN methods with similar bitrates

Method	# parameters	Theoretical bpd	Codeword bpd	Comp. time (s)	Decomp. time (s)
PC (HCLT, $M=16$)	3.3M	1.26	1.30	9	44
PC (HCLT, $M=24$)	5.1M	1.22	1.26	15	86
PC (HCLT, $M=32$)	7.0M	1.20	1.24	26	142
IDF	24.1M	1.90	1.96	288	592
BitSwap	2.8M	1.27	1.31	578	326

Anji Liu, Stephan Mandt and Guy Van den Broeck. Lossless Compression with Probabilistic Circuits, 2021.

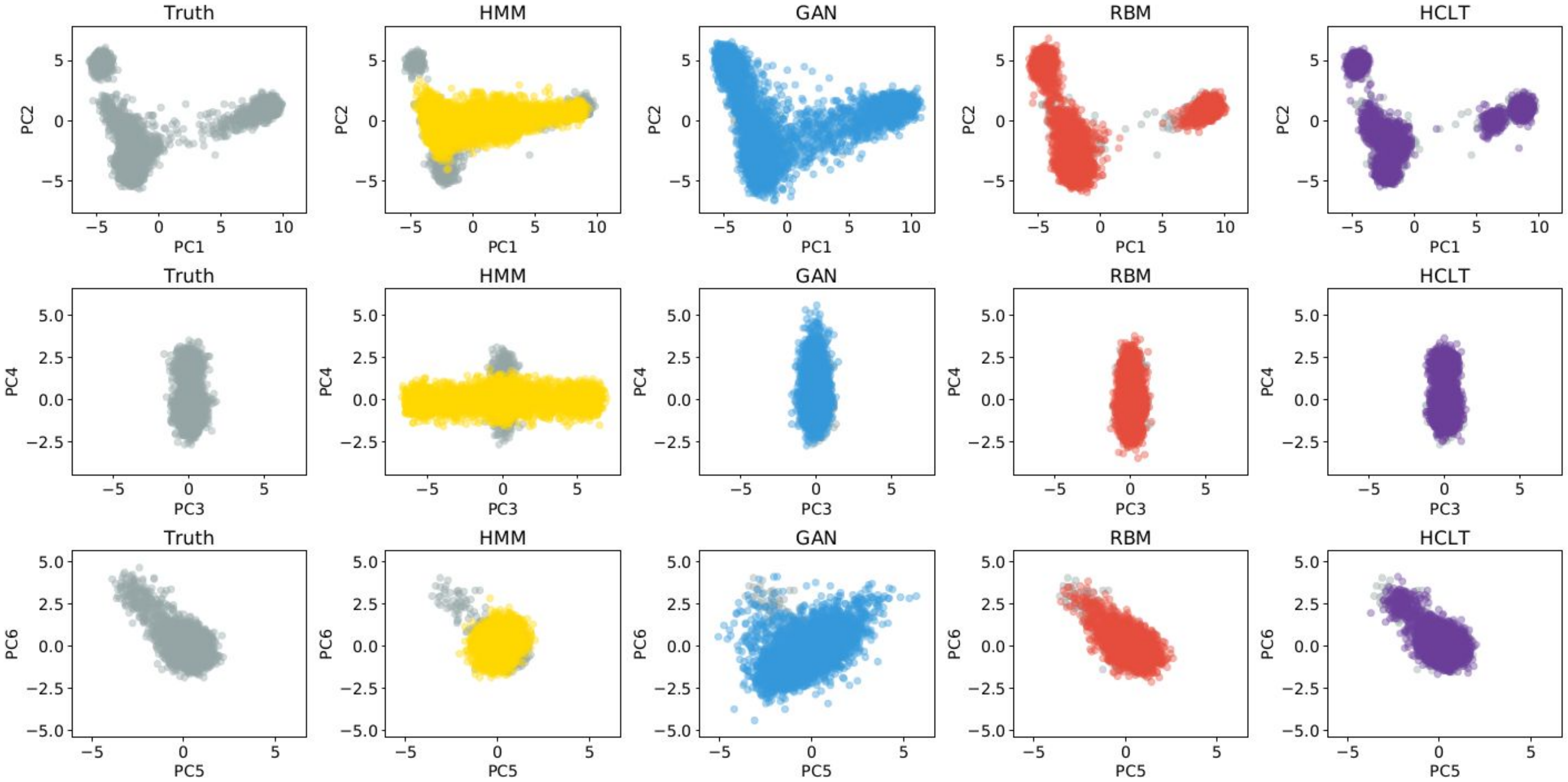
Lossless Neural Compression with Probabilistic Circuits

Can be effectively combined with Flow models to achieve better generative performance

Model	CIFAR10	ImageNet32	ImageNet64
RealNVP	3.49	4.28	3.98
Glow	3.35	4.09	3.81
IDF	3.32	4.15	3.90
IDF++	**3.24**	4.10	3.81
PC+IDF	3.28	**3.99**	**3.71**

Anji Liu, Stephan Mandt and Guy Van den Broeck. Lossless Compression with Probabilistic Circuits, 2021.

Tractable and expressive generative models of genetic variation data



Meihua Dang , Anji Liu , Xinzhu Wei , Sriram Sankararaman, and Guy Van den Broeck, Tractable and expressive generative models of genetic variation data, RECOMB 2022

PC Learners keep getting better! ***… stay tuned …***

Table 1: Density estimation performance on MNIST-family datasets in test set bpd.

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

Dataset	PC	Bipartite flow	AF/SCF	IAF/SCF
Penn Treebank	**1.23**	1.38	1.46	1.63

Meihua Dang, Anji Liu, Guy Van den Broeck, Sparse Probabilistic Circuits via Pruning and Growing, Sparsity in Neural Networks, 2022

Fully factorized

NaiveBayes *AndOrGraphs* *PDGs*

Trees *PSDDs* *CNets* *LTMs* *SPNs* *NADEs*

Thin Junction Trees *ACs* *MADEs* *MAFs* *VAEs*

DPPs *FVSBNs* *TACs* *IAFs* *NAFs* *RAEs*

Mixtures *BNs* *NICE* *FGs* *GANs*

RealNVP *MNs*

Expressive models without *compromises*

Smoothness + **decomposability** = ~~**tractable MAP**~~

We ***cannot*** decompose bottom-up a MAP query:

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

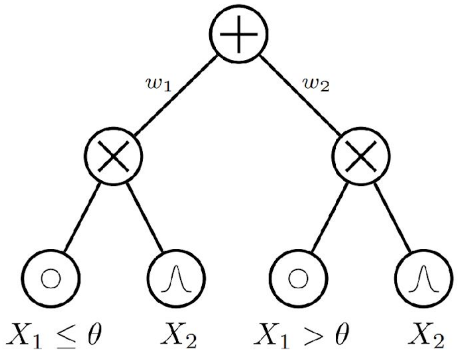
since for a sum node we are marginalizing out a latent variable

$$\max_{\mathbf{q}} \sum_{i} w_i p_i(\mathbf{q}, \mathbf{e}) = \max_{\mathbf{q}} \sum_{\mathbf{z}} p(\mathbf{q}, \mathbf{z}, \mathbf{e}) \neq \sum_{\mathbf{z}} \max_{\mathbf{q}} p(\mathbf{q}, \mathbf{z}, \mathbf{e})$$

$\Longrightarrow$ *MAP for latent variable models is* **intractable** *[Conaty et al. 2017]*

Determinism

A sum node is ***deterministic*** if only one of its children outputs non-zero for any input



deterministic circuit

$\Rightarrow$ *allows **tractable MAP** inference*

$$argmax_{\boldsymbol{x}}\, p(\boldsymbol{x})$$

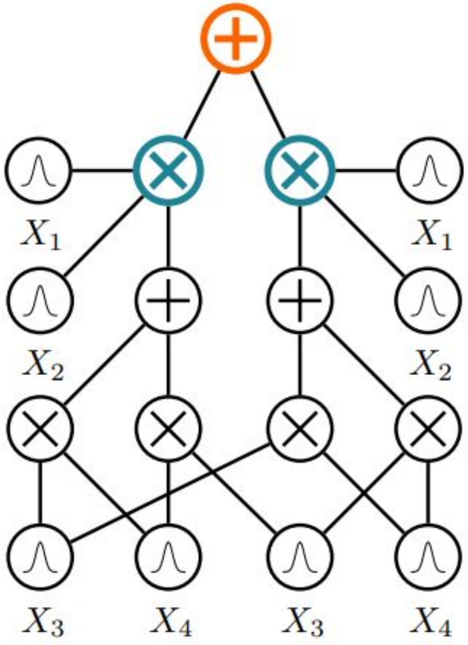
Darwiche and Marquis, “A Knowledge Compilation Map”, 2002

Determinism + decomposability = tractable MAP

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

$$\max_{\mathbf{q}} p(\mathbf{q}, \mathbf{e}) = \max_{\mathbf{q}} \sum_i w_i p_i(\mathbf{q}, \mathbf{e})$$
$$= \max_{\mathbf{q}} \max_i w_i p_i(\mathbf{q}, \mathbf{e})$$
$$= \max_i \max_{\mathbf{q}} w_i p_i(\mathbf{q}, \mathbf{e})$$

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

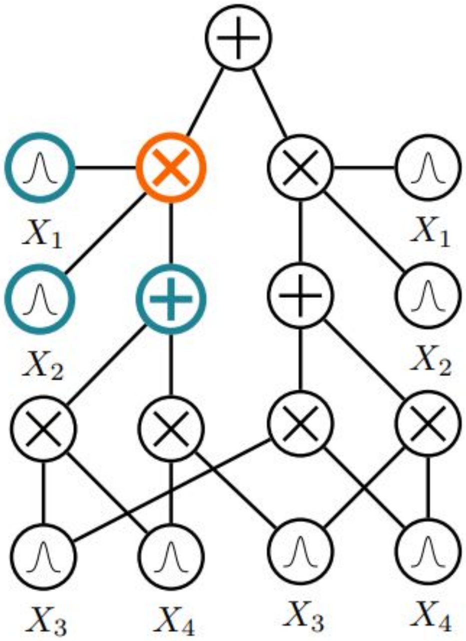


Determinism + **decomposability** = **tractable MAP**

If $p(\mathbf{q}, \mathbf{e}) = p(\mathbf{q}_\mathbf{x}, \mathbf{e}_\mathbf{x}, \mathbf{q}_\mathbf{y}, \mathbf{e}_\mathbf{y}) = p(\mathbf{q}_\mathbf{x}, \mathbf{e}_\mathbf{x}) p(\mathbf{q}_\mathbf{y}, \mathbf{e}_\mathbf{y})$
(***decomposable*** product node):

$$
\begin{aligned}
\max_{\mathbf{q}} p(\mathbf{q} \mid \mathbf{e}) &= \max_{\mathbf{q}} p(\mathbf{q}, \mathbf{e}) \\
&= \max_{\mathbf{q}_\mathbf{x}, \mathbf{q}_\mathbf{y}} p(\mathbf{q}_\mathbf{x}, \mathbf{e}_\mathbf{x}, \mathbf{q}_\mathbf{y}, \mathbf{e}_\mathbf{y}) \\
&= \max_{\mathbf{q}_\mathbf{x}} p(\mathbf{q}_\mathbf{x}, \mathbf{e}_\mathbf{x}) \cdot \max_{\mathbf{q}_\mathbf{y}} p(\mathbf{q}_\mathbf{y}, \mathbf{e}_\mathbf{y})
\end{aligned}
$$

$\Longrightarrow$ *solving optimization independently*



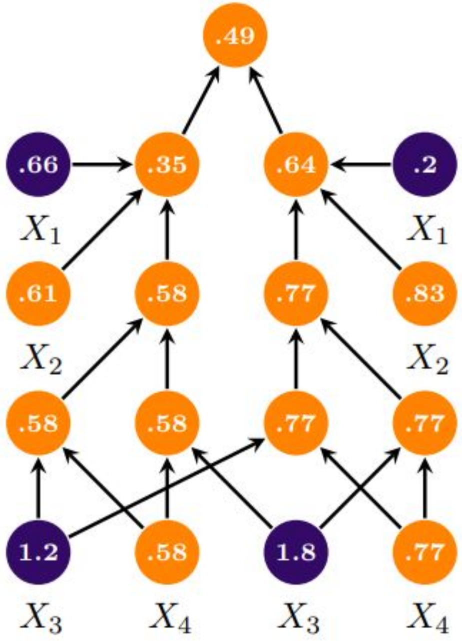
Determinism + decomposability = tractable MAP

Evaluating the circuit twice:

bottom-up and ***top-down*** $\Rightarrow$ *still linear in circuit size!*

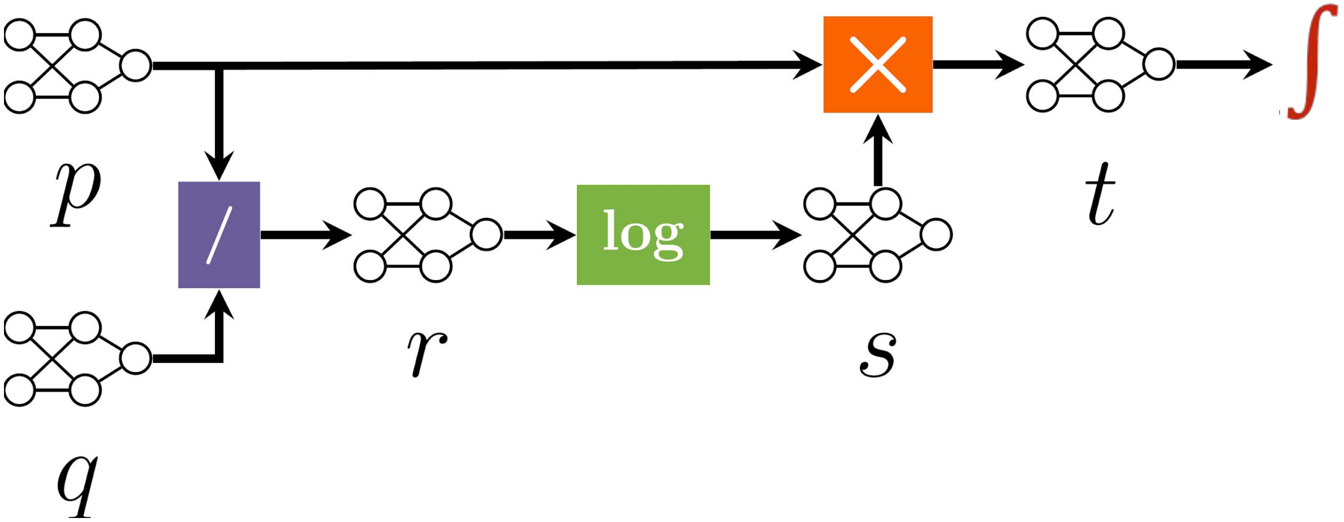
E.g., for $\text{argmax}_{x_1, x_3}\, p(x_1, x_3 \mid x_2, x_4)$:

1. turn sum into max nodes and distributions into max distributions
2. evaluate $p(x_2, x_4)$ bottom-up
3. retrieve max activations top-down
4. compute **MAP states** for X_1 and X_3 at leaves



Queries as pipelines: KLD

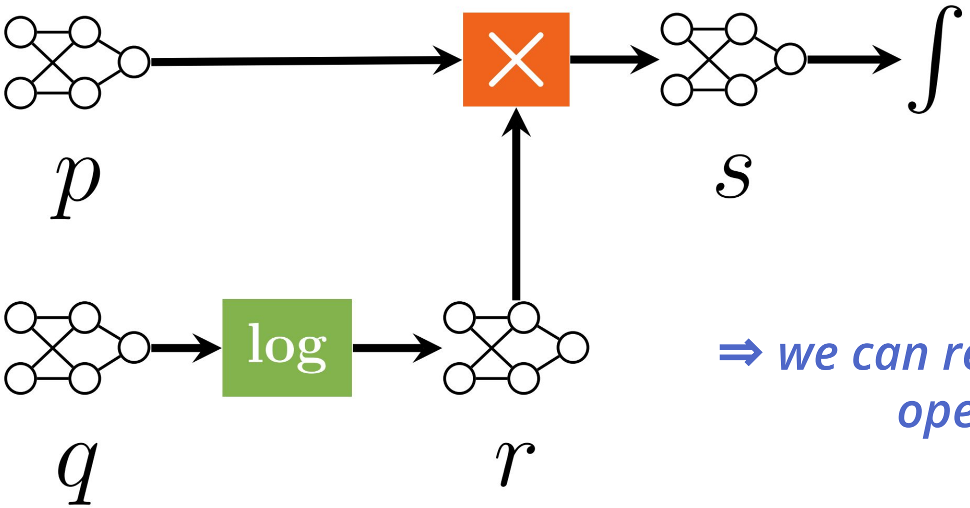
$$\mathbb{KLD}(p \parallel q) = \int p(\boldsymbol{x}) \times \log((p(\boldsymbol{x})/q(\boldsymbol{x}))d\boldsymbol{X}$$



Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso and Guy Van den Broeck. A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference, *NeurIPS*, 2021.

Queries as pipelines: Cross Entropy

$$H(p,q) = \int p(\boldsymbol{x}) \times \log(q(\boldsymbol{x}))d\boldsymbol{X}$$



⇒ *we can reuse the operations!*

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso and Guy Van den Broeck. A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference, *NeurIPS*, 2021.

Operation		Tractability	
		Input conditions	Output conditions
LOG	$\log(p)$	Sm, Dec, Det	Sm, Dec

smooth,
decomposable,
deterministic

smooth,
decomposable

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso and Guy Van den Broeck. A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference, *NeurIPS*, 2021.

Tractable circuit operations

Operation		Tractability: Input properties	Tractability: Output properties	Hardness
SUM	$\theta_1 p + \theta_2 q$	(+Cmp)	(+SD)	NP-hard for Det output
PRODUCT	$p \cdot q$	Cmp (+Det, +SD)	Dec (+Det, +SD)	#P-hard w/o Cmp
POWER	$p^n, n \in \mathbb{N}$	SD (+Det)	SD (+Det)	#P-hard w/o SD
	$p^\alpha, \alpha \in \mathbb{R}$	Sm, Dec, Det (+SD)	Sm, Dec, Det (+SD)	#P-hard w/o Det
QUOTIENT	p/q	Cmp; q Det (+p Det,+SD)	Dec (+Det,+SD)	#P-hard w/o Det
LOG	$\log(p)$	Sm, Dec, Det	Sm, Dec	#P-hard w/o Det
EXP	$\exp(p)$	linear	SD	#P-hard

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso and Guy Van den Broeck. A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference, *NeurIPS*, 2021.

Inference by tractable operations

systematically derive tractable inference algorithm of complex queries

	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

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso and Guy Van den Broeck. A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference, *NeurIPS*, 2021.

Even harder queries

Marginal MAP

Given a set of query variables $\boldsymbol{Q} \subset \boldsymbol{X}$ and evidence $\boldsymbol{e}$,

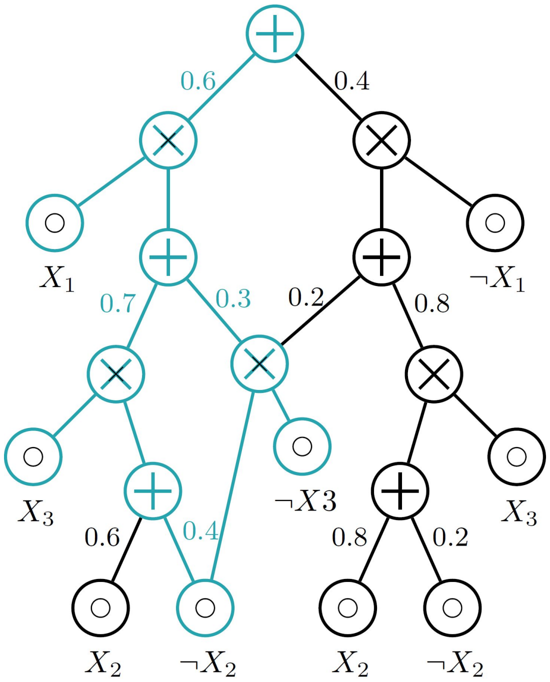
find: $argmax_{\boldsymbol{q}}\, p(\boldsymbol{q}|\boldsymbol{e})$

$\Rightarrow$ *i.e. MAP of a marginal distribution on* ***Q***

! ***NP^{PP}-complete*** *for PGMs*

! ***NP-hard*** *even for PCs tractable for marginals, MAP & entropy*

Pruning circuits



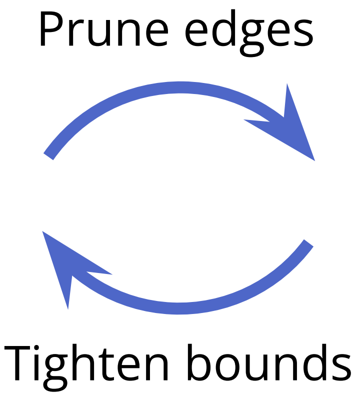
Any parts of circuit not relevant for MMAP state can be pruned away

e.g. $p(X_1 = 1, X_2 = 0)$

We can find such edges in *linear time*

YooJung Choi, Tal Friedman and Guy Van den Broeck. Solving Marginal MAP Exactly by Probabilistic Circuit Transformations, 2021.

Iterative MMAP solver



Dataset	runtime (# solved) search	runtime (# solved) pruning
NLTCS	**0.01** (10)	0.63 (10)
MSNBC	**0.03** (10)	0.73 (10)
KDD	**0.04** (10)	0.68 (10)
Plants	2.95 (10)	**2.72** (10)
Audio	2041.33 (6)	**13.70 (10)**
Jester	2913.04 (2)	**14.74 (10)**
Netflix	– (0)	**47.18 (10)**
Accidents	109.56 (10)	**15.86** (10)
Retail	**0.06** (10)	0.81 (10)
Pumsb-star	2208.27 (7)	**20.88 (10)**
DNA	– (0)	**505.75 (9)**
Kosarek	48.74 (10)	**3.41** (10)
MSWeb	1543.49 (10)	**1.28** (10)
Book	– (0)	**46.50 (10)**
EachMovie	– (0)	**1216.89 (8)**
WebKB	– (0)	**575.68 (10)**
Reuters-52	– (0)	**120.58 (10)**
20 NewsGrp.	– (0)	**504.52 (9)**
BBC	– (0)	**2757.18 (3)**
Ad	– (0)	**1254.37 (8)**

YooJung Choi, Tal Friedman and Guy Van den Broeck. Solving Marginal MAP Exactly by Probabilistic Circuit Transformations, 2021.

Probabilistic Sufficient Explanations

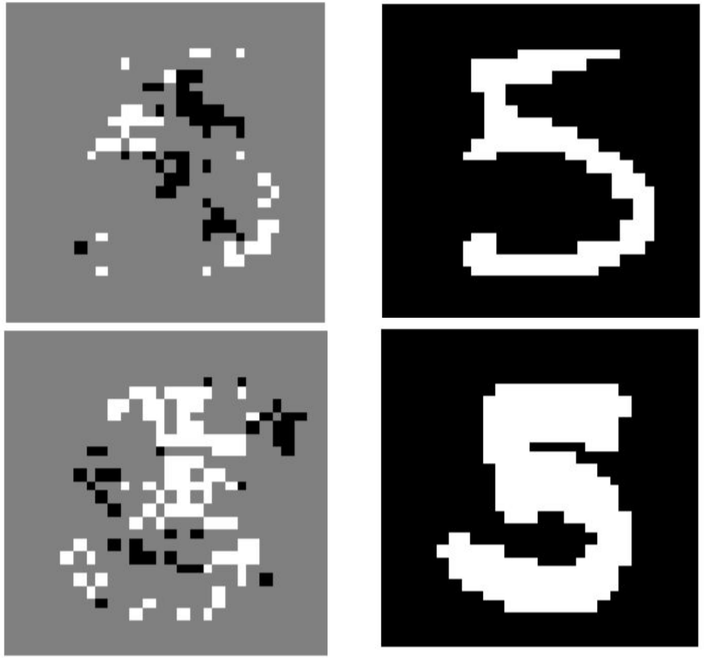
<u>Goal</u>: explain an instance of classification (a specific prediction)

Explanation is a subset of features, s.t.

1. The explanation is "probabilistically sufficient"

   Under the feature distribution, given the explanation, the classifier is likely to make the observed prediction.

2. It is minimal and "simple"



[Khosravi et al. IJCAI19, Wang et al. XXAI20]

Model-Based Algorithmic Fairness: FairPC

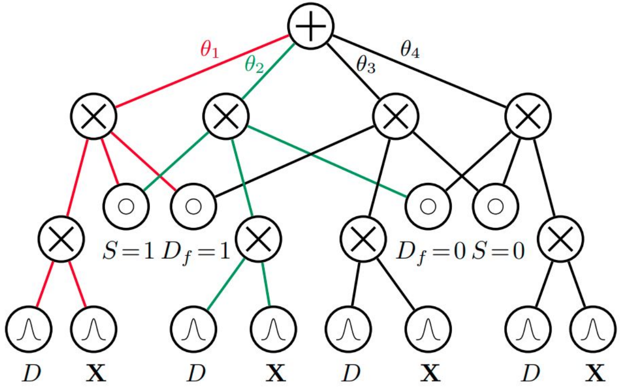
Learn classifier given

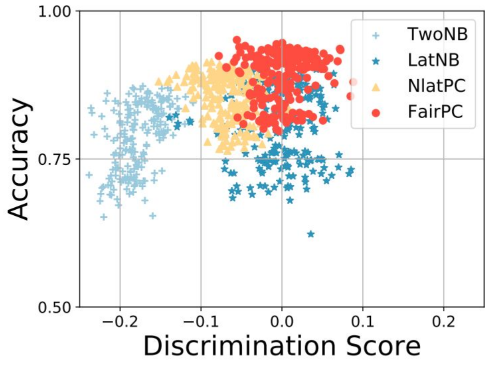
- features S and X
- training labels/decisions D

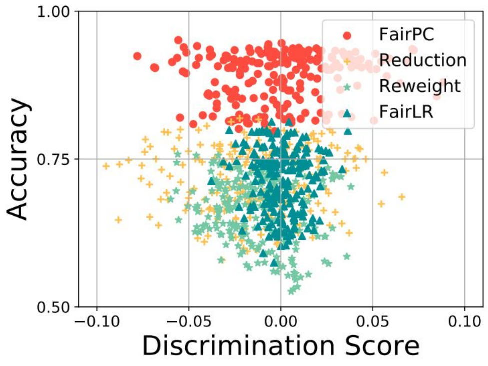
Group fairness by demographic parity:

Fair decision D_f should be independent of the sensitive attribute S

Discover the **latent fair decision** D_f by learning a PC.

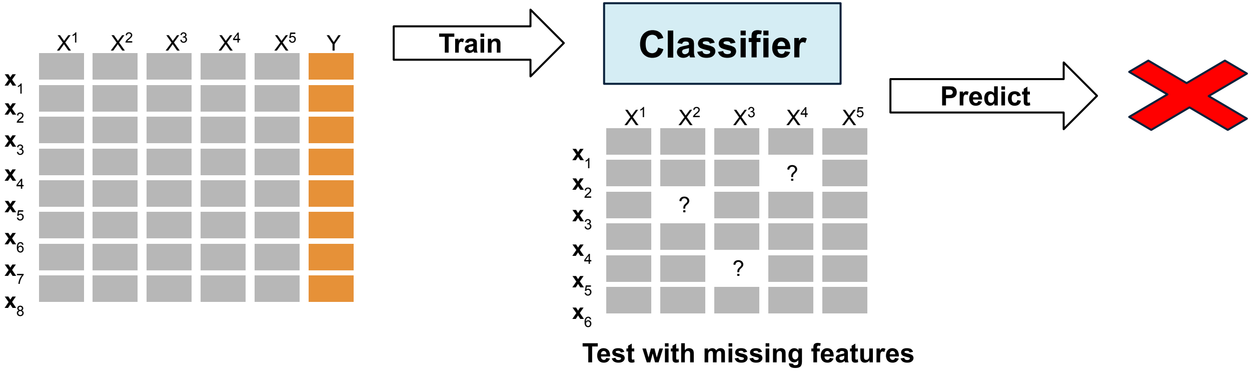






[Choi et al. AAAI21]

Prediction with Missing Features



See work on

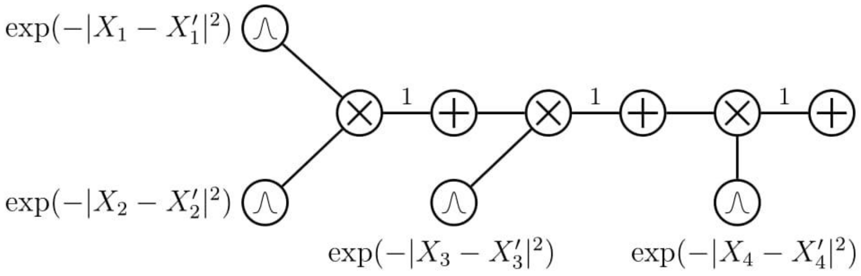
- Expected predictions / conformant learning [Khosravi et al.]
- Generative forests [Correia et al.]

Tractable Computation of Expected Kernels

- How to compute the expected kernel given two distributions **p**, **q**?

$$\mathbb{E}_{\mathbf{x}\sim\mathbf{p},\mathbf{x}'\sim\mathbf{q}}[\mathbf{k}(\mathbf{x},\mathbf{x}')]$$

- Circuit representation for kernel functions, e.g., $\mathbf{k}(\mathbf{x},\mathbf{x}') = \exp\left(-\sum_{i=1}^{4} \mid X_i - X_i' \mid^2\right)$



Wenzhe Li, Zhe Zeng, Antonio Vergari and Guy Van den Broeck. Tractable Computation of Expected Kernels, *UAI*, 2021.

Tractable Computation of Expected Kernels: Applications

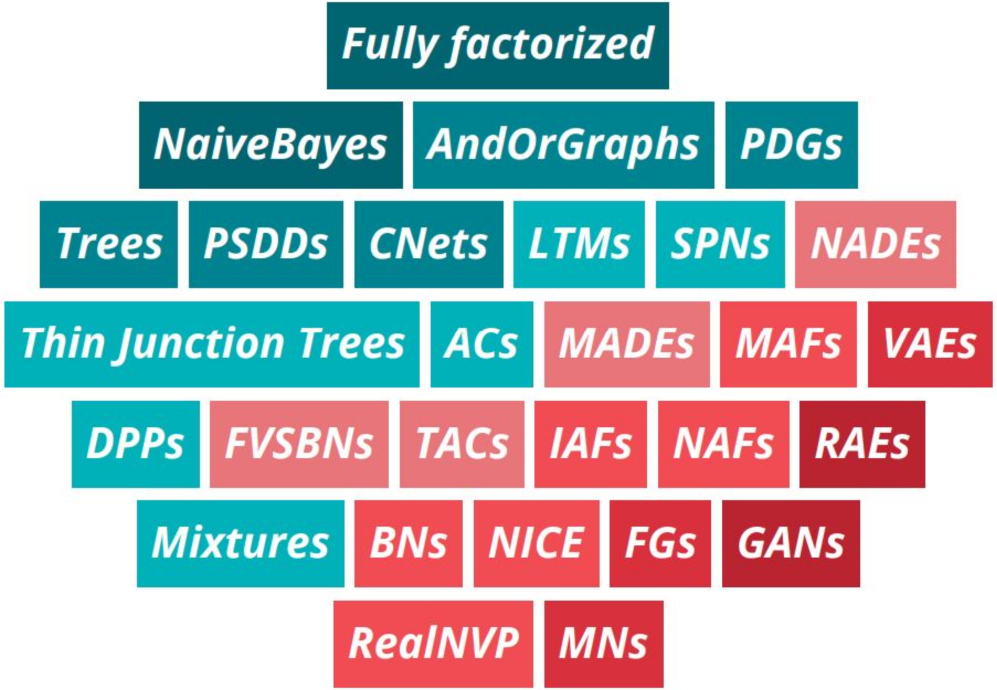
- Reasoning about support vector regression (SVR) with missing features

$$\underbrace{\mathbb{E}_{\mathbf{x}_m \sim \mathbf{p}(\mathbf{X}_m|\mathbf{x}_o)}}_{\text{missing features}} \underbrace{\left[\sum_{i=1}^{m} w_i \mathbf{k}(\mathbf{x}_i, \mathbf{x}) + b\right]}_{\text{SVR model}}$$

- Collapsed Black-box Importance Sampling: minimize kernelized Stein discrepancy

importance weights $\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmin}} \left\{ \boldsymbol{w}^\top \underbrace{\boldsymbol{K}_{p,\mathbf{s}}}_{\text{expected kernel matrix}} \boldsymbol{w} \,\middle|\, \sum_{i=1}^{n} w_i = 1,\ w_i \geq 0 \right\}$

Wenzhe Li, Zhe Zeng, Antonio Vergari and Guy Van den Broeck. Tractable Computation of Expected Kernels, *UAI*, 2021.



tractability is a spectrum

Probabilistic circuits seem awfully general.

Are all tractable probabilistic models probabilistic circuits?

Enter: Determinantal Point Processes (DPPs)

DPPs are models where probabilities are specified by (sub)determinants

$$L = \begin{bmatrix} 1 & 0.9 & 0.8 & 0 \\ 0.9 & 0.97 & 0.96 & 0 \\ 0.8 & 0.96 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

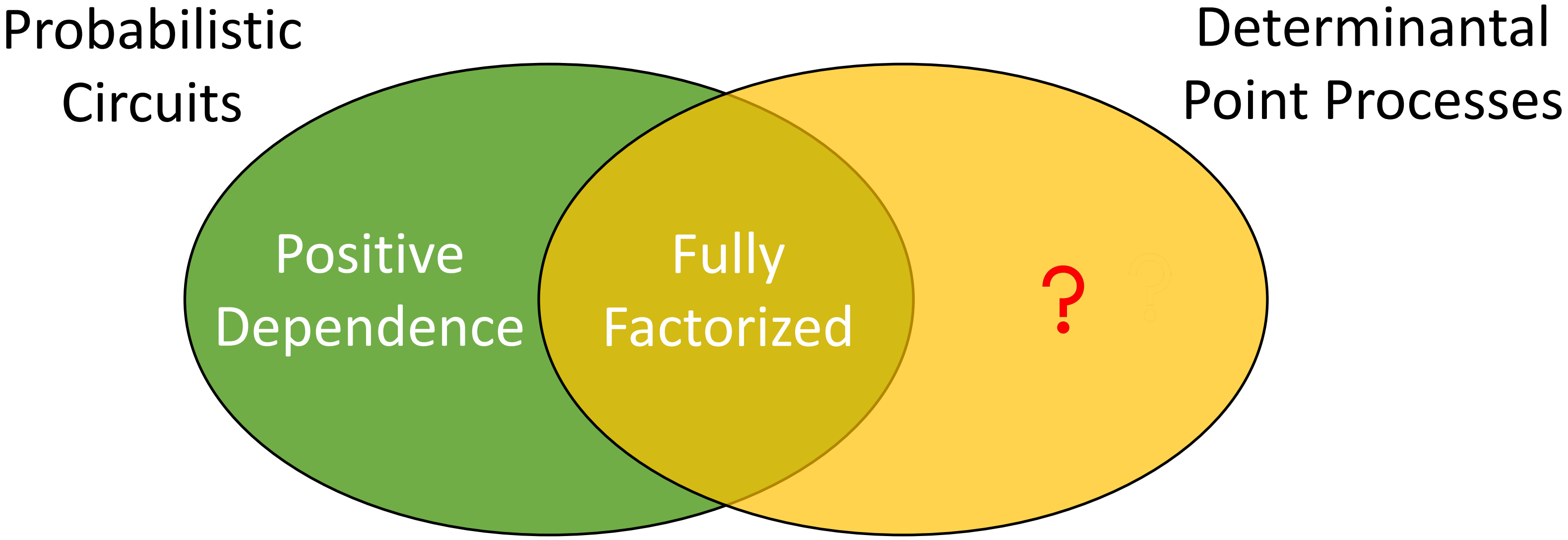
Tractable likelihoods and marginals

Global Negative Dependence

Diversity in recommendation systems

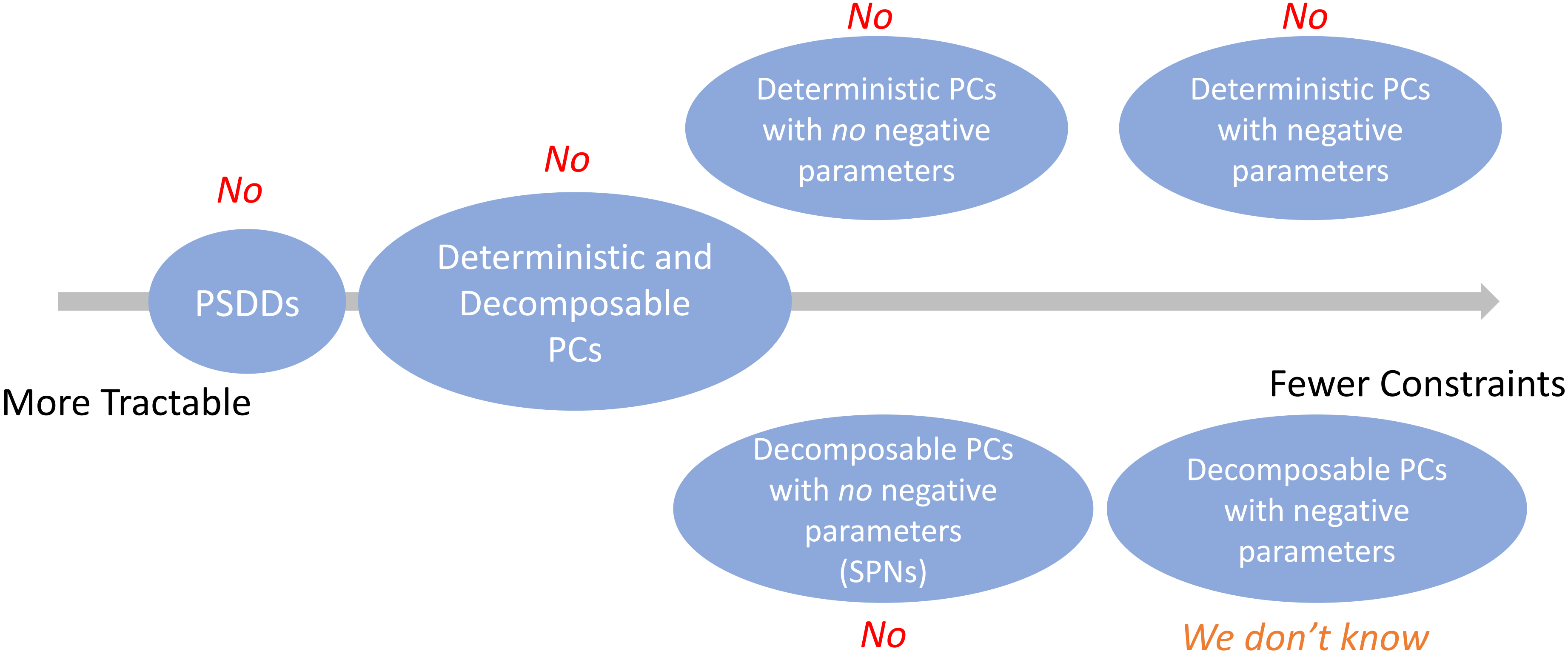
$$\Pr_L(X_1 = 1, X_2 = 0, X_3 = 1, X_4 = 0) = \frac{1}{\det(L + I)} \det(L_{\{1,2\}})$$

Relationship between PCs and DPPs



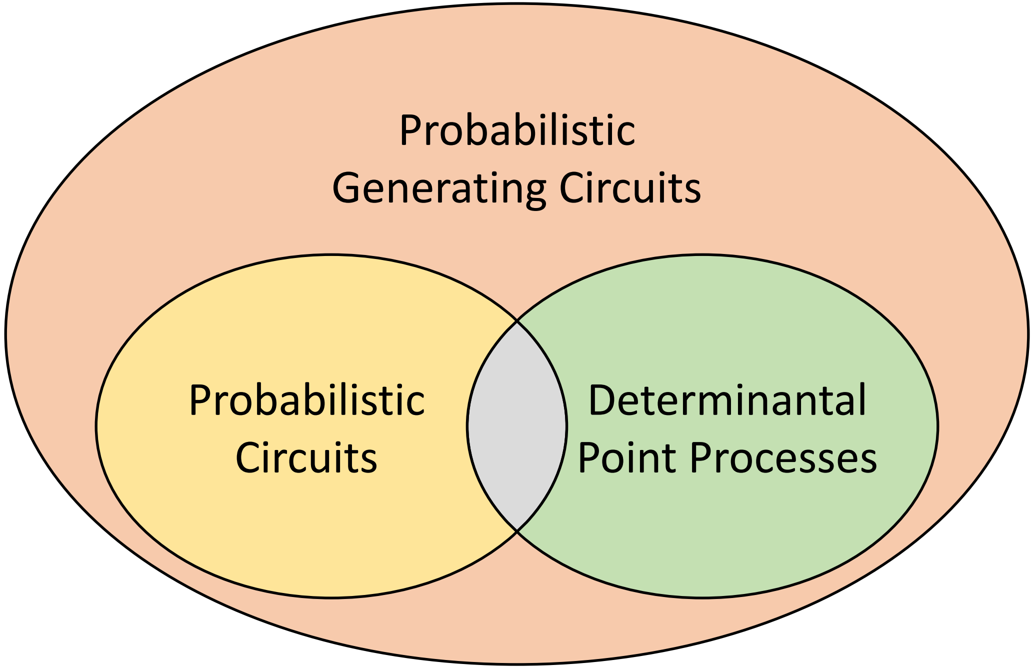
Honghua Zhang, Steven Holtzen and Guy Van den Broeck. On the Relationship Between Probabilistic Circuits and Determinantal Point Processes, *UAI*, 2020.

We cannot tractably represent DPPs with subclasses of PCs



Honghua Zhang, Steven Holtzen and Guy Van den Broeck. On the Relationship Between Probabilistic Circuits and Determinantal Point Processes, *UAI*, 2020.

Probabilistic Generating Circuits



A Tractable Unifying Framework for PCs and DPPs

Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

Probability Generating Functions

X_1	X_2	X_3	$\Pr_\beta$
0	0	0	0.02
0	0	1	0.08
0	1	0	0.12
0	1	1	0.48
1	0	0	0.02
1	0	1	0.08
1	1	0	0.04
1	1	1	0.16

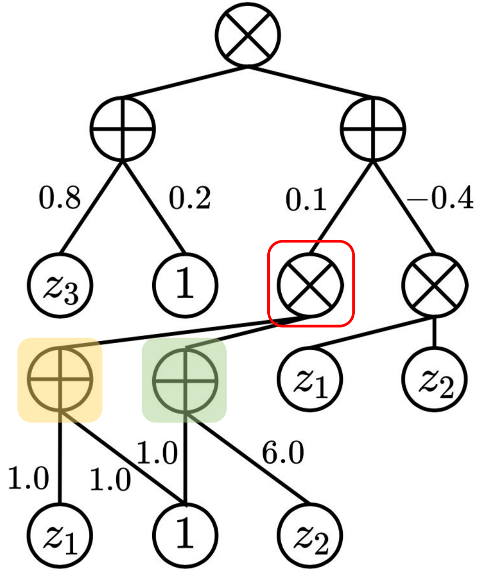
$$g_\beta = \boxed{0.16z_1z_2z_3} + 0.04z_1z_2 + 0.08z_1z_3 + 0.02z_1 + 0.48z_2z_3 + 0.12z_2 + 0.08z_3 + 0.02.$$

$$g_\beta = (0.1(z_1+1)(6z_2+1) - 0.4z_1z_2)(0.8z_3+0.2)$$

Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

Probabilistic Generating Circuits (PGCs)

$$g_\beta = (0.1\boxed{(z_1+1)(6z_2+1)} - 0.4z_1z_2)(0.8z_3+0.2)$$



1. Sum nodes ⊕ with weighted edges to children.
2. Product nodes ⊗ with unweighted edges to children.
3. Leaf nodes: z_i or constant.

Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

DPPs as PGCs

The generating polynomial for a DPP with kernel L is given by:

$$g_L = \frac{1}{\det(L+I)} \det(I + L\mathrm{diag}(z_1, \ldots, z_n)).$$

Constant

Division-free determinant algorithm
(Samuelson-Berkowitz algorithm)

g_L can be represented as a PGC of size $O(n^4)$

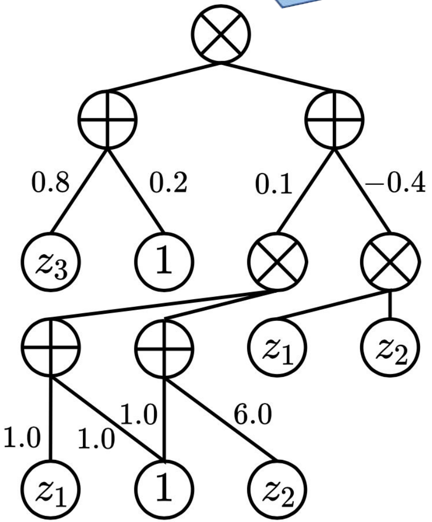
Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

PGCs Support Tractable Likelihoods/Marginals

Purely symbolic

$$z_i = \begin{cases} t, & X_i = 1 \\ 0, & X_i = 0 \\ 1, & otherwise \end{cases}$$

$\Pr(X_1 = 1, X_2 = 0, \ldots) = ?$



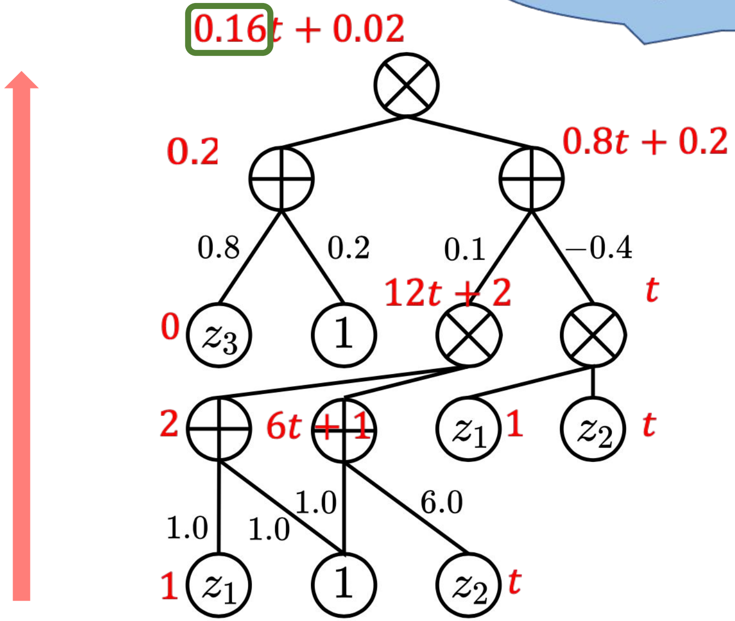
$p(t) = \alpha_k t^k + \cdots + \alpha_1 t$

α_k gives the answer

Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

Example

$\Pr(X_2 = 1, X_3 = 0) = ?$



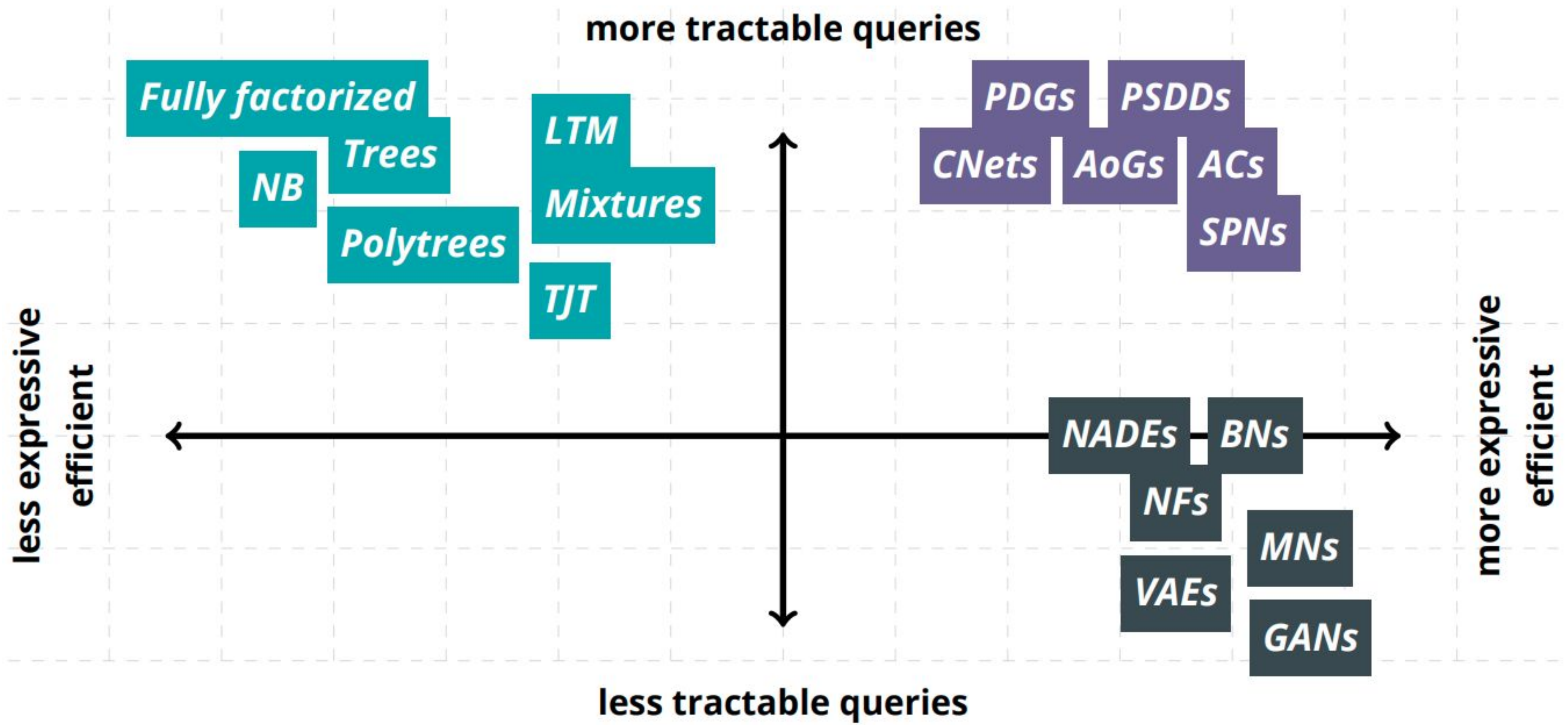
X_1	X_2	X_3	$\Pr_\beta$
0	0	0	0.02
0	0	1	0.08
0	1	0	0.12
0	1	1	0.48
1	0	0	0.02
1	0	1	0.08
1	1	0	0.04
1	1	1	0.16

Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

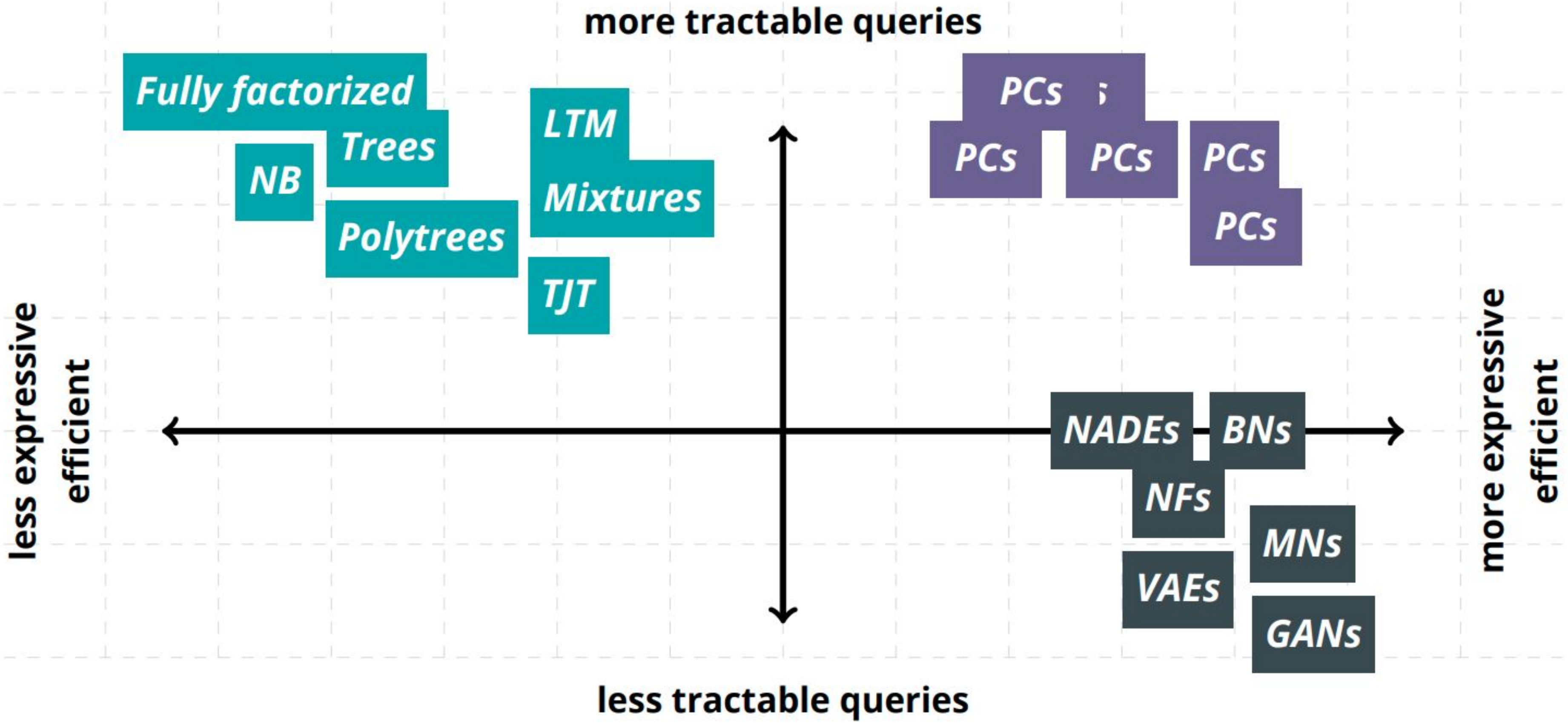
Experiment Results: Amazon Baby Registries

	DPP	Strudel	EiNet	MT	SimplePGC
apparel	-9.88	-9.51	-9.24	-9.31	$\mathbf{-9.10}^{*\dagger\circ}$
bath	-8.55	-8.38	-8.49	-8.53	$\mathbf{-8.29}^{*\dagger\circ}$
bedding	-8.65	-8.50	-8.55	-8.59	$\mathbf{-8.41}^{*\dagger\circ}$
carseats	-4.74	-4.79	-4.72	-4.76	$\mathbf{-4.64}^{*\dagger\circ}$
diaper	-10.61	-9.90	-9.86	-9.93	$\mathbf{-9.72}^{*\dagger\circ}$
feeding	-11.86	-11.42	-11.27	-11.30	$\mathbf{-11.17}^{*\dagger\circ}$
furniture	-4.38	-4.39	-4.38	-4.43	$\mathbf{-4.34}^{*\dagger\circ}$
gear	-9.14	-9.15	-9.18	-9.23	$\mathbf{-9.04}^{*\dagger\circ}$
gifts	-3.51	$\mathbf{-3.39}$	-3.42	-3.48	-3.47°
health	-7.40	-7.37	-7.47	-7.49	$\mathbf{-7.24}^{*\dagger\circ}$
media	-8.36	$\mathbf{-7.62}$	-7.82	-7.93	$-7.69^{\dagger\circ}$
moms	-3.55	-3.52	$\mathbf{-3.48}$	-3.54	-3.53°
safety	-4.28	-4.43	-4.39	-4.36	$\mathbf{-4.28}^{*\dagger\circ}$
strollers	-5.30	-5.07	-5.07	-5.14	$\mathbf{-5.00}^{*\dagger\circ}$
toys	-8.05	$\mathbf{-7.61}$	-7.84	-7.88	$-7.62^{\dagger\circ}$

SimplePGC achieves SOTA result on 11/15 datasets



more tractable queries
Fully factorized
Trees
NB
LTM
Mixtures
Polytrees
TJT
PDGs
PSDDs
CNets
AoGs
ACs
SPNs
less expressive
efficient
more expressive
efficient
NADEs
BNs
NFs
MNs
VAEs
GANs
less tractable queries

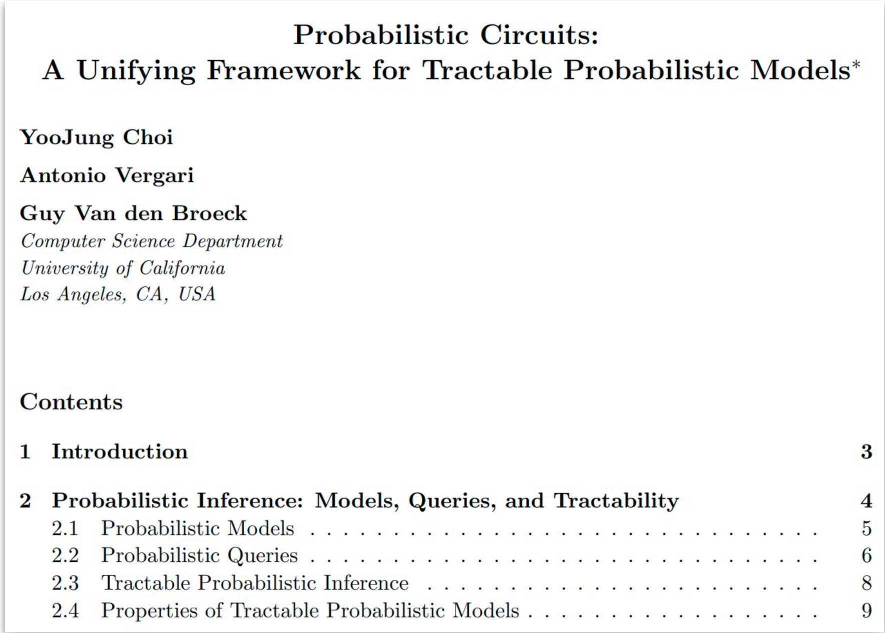

more tractable queries
Fully factorized
Trees
NB
Polytrees
LTM
Mixtures
TJT
PCs
PCs
PCs
PCs
PCs
less expressive
efficient
more expressive
efficient
NADEs
BNs
NFs
MNs
VAEs
GANs
less tractable queries

Learn more about probabilistic circuits?

Tutorial (3h)

https://youtu.be/2RAG5-L9R70

Overview Paper (80p)



Probabilistic Circuits:
A Unifying Framework for Tractable Probabilistic Models*

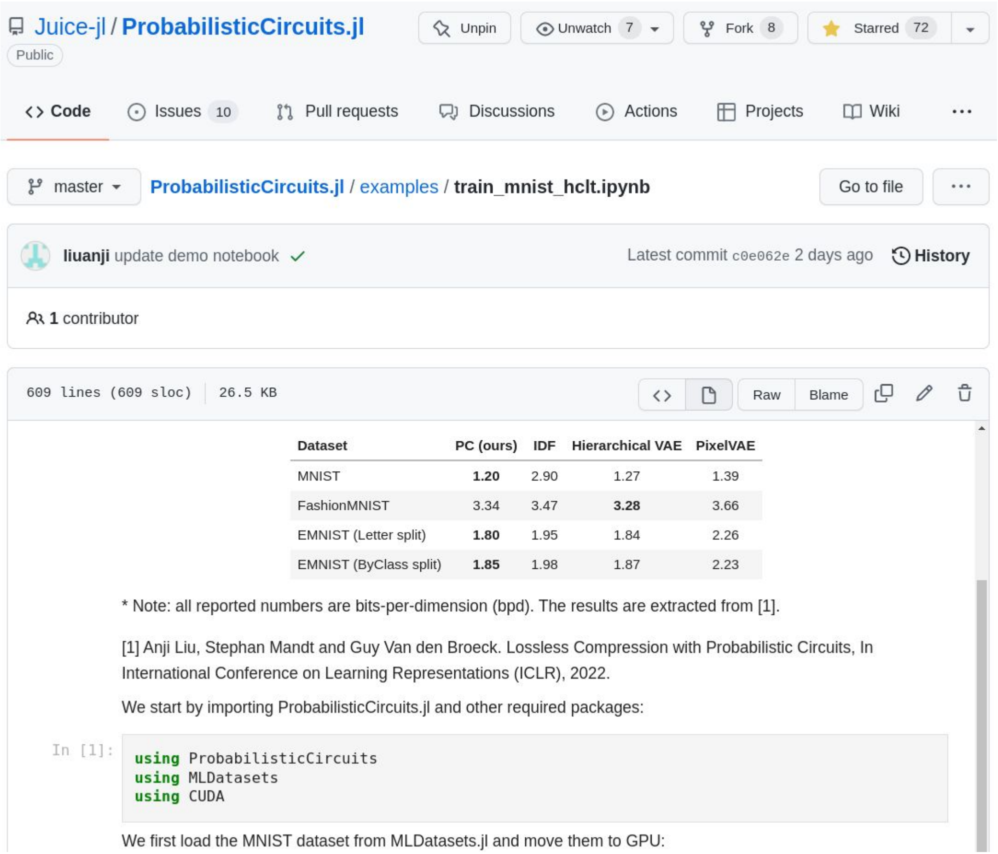
YooJung Choi
Antonio Vergari
Guy Van den Broeck
Computer Science Department
University of California
Los Angeles, CA, USA

Contents

http://starai.cs.ucla.edu/papers/ProbCirc20.pdf

Training SotA likelihood full MNIST probabilistic circuit model in ~7 minutes on GPU:

https://github.com/Juice-jl/ProbabilisticCircuits.jl/blob/master/examples/train_mnist_hclt.ipynb



Dataset	PC (ours)	IDF	Hierarchical VAE	PixelVAE
MNIST	**1.20**	2.90	1.27	1.39
FashionMNIST	3.34	3.47	**3.28**	3.66
EMNIST (Letter split)	**1.80**	1.95	1.84	2.26
EMNIST (ByClass split)	**1.85**	1.98	1.87	2.23

* Note: all reported numbers are bits-per-dimension (bpd). The results are extracted from [1].

[1] Anji Liu, Stephan Mandt and Guy Van den Broeck. Lossless Compression with Probabilistic Circuits, In International Conference on Learning Representations (ICLR), 2022.

We start by importing ProbabilisticCircuits.jl and other required packages:

In [1]:
```
using ProbabilisticCircuits
using MLDatasets
using CUDA
```

We first load the MNIST dataset from MLDatasets.jl and move them to GPU:

Outline

1. The paradox of learning to reason from data

~~deep learning~~

2. Tractable deep generative models

probabilistic reasoning + deep learning

3. **Learning with symbolic knowledge**

logical reasoning + deep learning

The AI Dilemma

Pure (Logic) Reasoning **Pure Learning**

- Slow thinking: deliberative, cognitive, model-based, extrapolation
- Amazing achievements until this day

- "*Pure logic is brittle*"

  noise, uncertainty, incomplete knowledge, …

The AI Dilemma

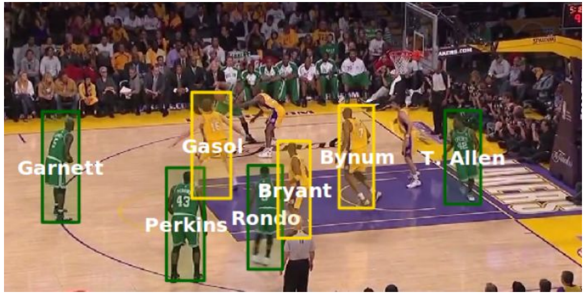
Pure (Logic) Reasoning **Pure Learning**

- Fast thinking: instinctive, perceptive, model-free, interpolation
- Amazing achievements recently
- “*Pure learning is brittle*”

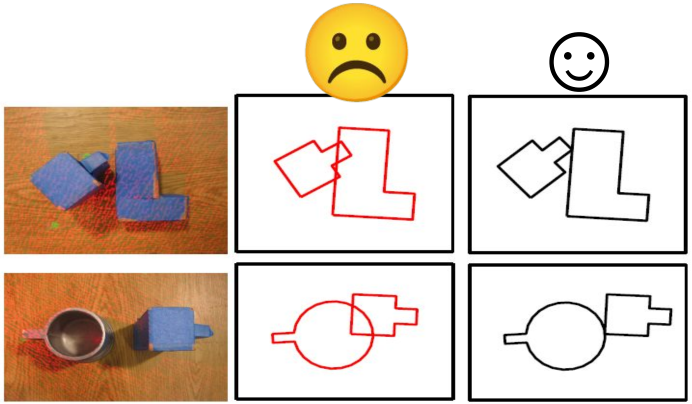
bias, algorithmic fairness, interpretability, explainability, adversarial attacks, unknown unknowns, calibration, verification, missing features, missing labels, data efficiency, shift in distribution, general robustness and safety

fails to incorporate a sensible model of the world

Knowledge in Vision, Robotics, NLP, Activity Recognition



People appear at most once in a frame



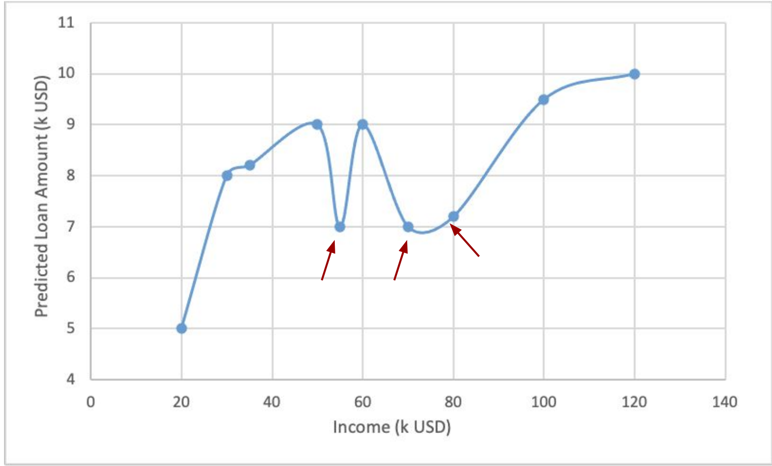
Rigid objects don't overlap

At least one verb in each sentence.
If X and Y are married, then they are people.

Cut the orange before squeezing the orange

[Lu, W. L., Ting, J. A., Little, J. J., & Murphy, K. P. (2013). Learning to track and identify players from broadcast sports videos.], [Wong, L. L., Kaelbling, L. P., & Lozano-Perez, T., Collision-free state estimation. ICRA 2012], [Chang, M., Ratinov, L., & Roth, D. (2008). Constraints as prior knowledge], [Ganchev, K., Gillenwater, J., & Taskar, B. (2010). Posterior regularization for structured latent variable models]… and many many more!

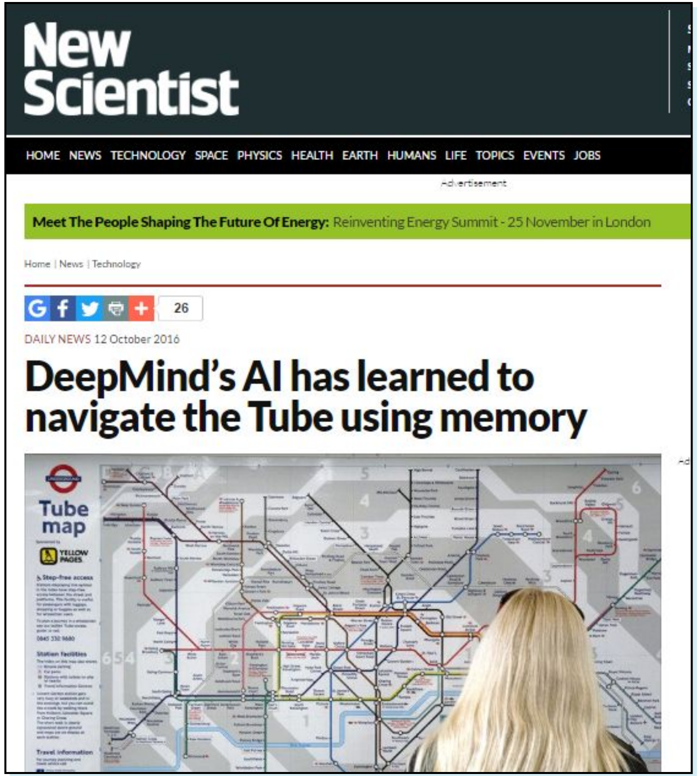
Predict Loan Amount

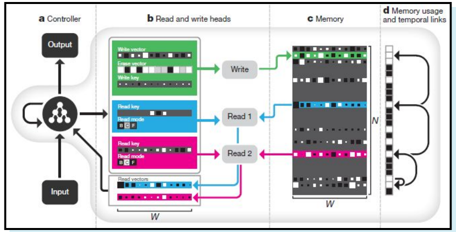


Neural Network Model: Increasing income can decrease the approved loan amount

Monotonicity (Prior Knowledge):
Increasing income should increase the approved loan amount

Motivation: Deep Learning



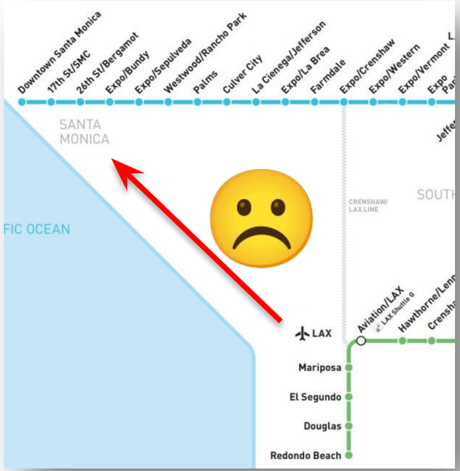


[Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., et al.. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, *538*(7626), 471-476.]

Motivation: Deep Learning

DeepMind's latest technique uses external memory to **solve tasks that require logic and reasoning — a step toward more human-like AI.**

… but …



optimal planner recalculating a shortest path to the end node. To ensure that the network always moved to a valid node, the output distribution was renormalized over the set of possible triples outgoing from the current node. The performance

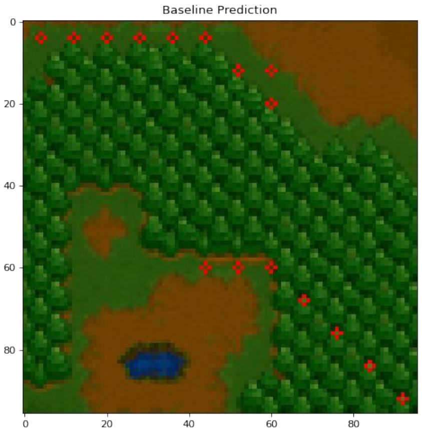
it also received input triples during the answer phase, indicating the actions chosen on the previous time-step. This makes the problem a 'structured prediction'

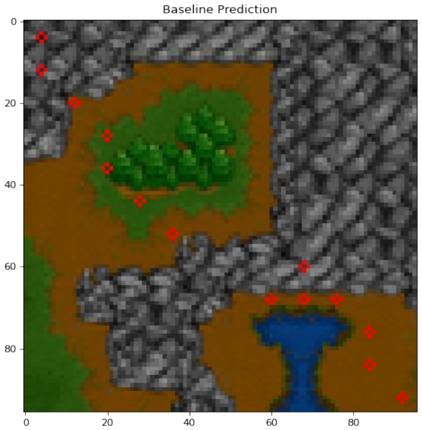
[Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., et al.. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, *538*(7626), 471-476.]

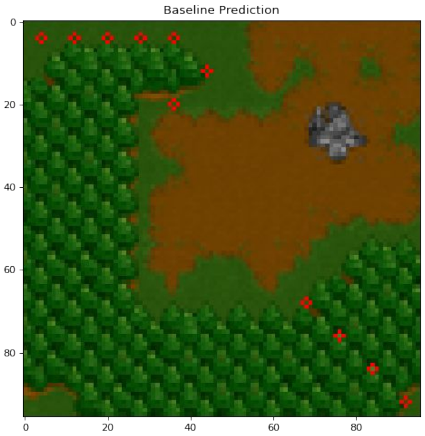
Warcraft Shortest Path

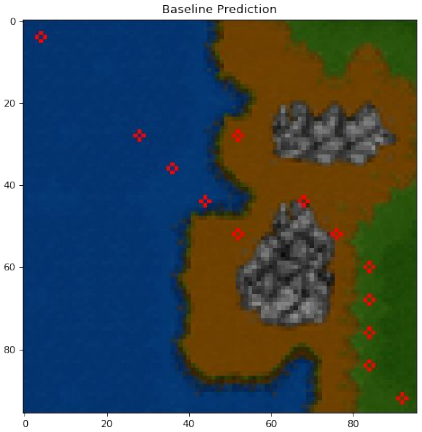
Predicting the minimum-cost path

[Differentiation of Blackbox Combinatorial Solvers, Marin Vlastelica, Anselm Paulus, Vít Musil, Georg Martius, Michal Rolínek, 2019]


Baseline Prediction
0
20
40
60
80
0
20
40
60
80


Baseline Prediction
0
20
40
60
80
0
20
40
60
80

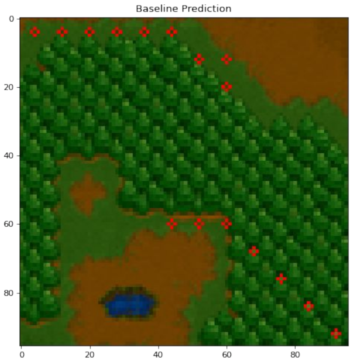

Baseline Prediction
0
20
40
60
80
0
20
40
60
80

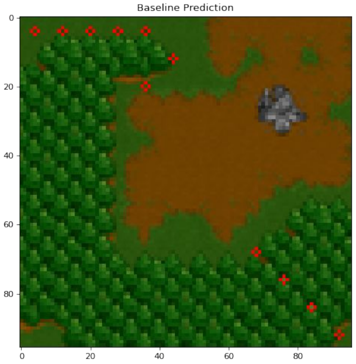

Baseline Prediction
0
20
40
60
80
0
20
40
60
80

Knowledge vs. Data

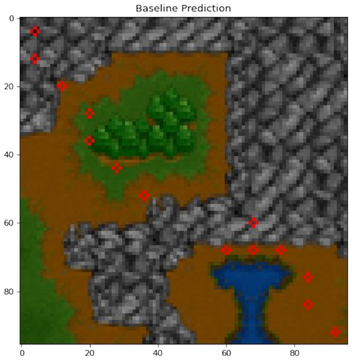
- Where did the world knowledge go?
  - Python scripts
    - Decode/encode/search cleverly
    - Fix inconsistent beliefs
  - Rule-based decision systems
  - Dataset design
  - “a big hack” (with author’s permission)
- In some sense we went backwards

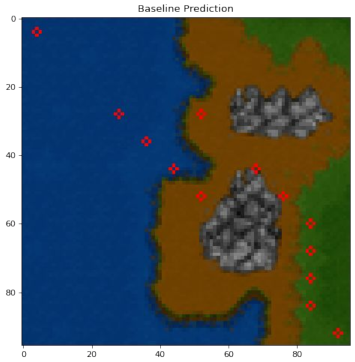
  Less principled, scientific, and intellectually satisfying ways of incorporating knowledge *without constraint*





without constraint





Warcraft min-cost simple-path prediction results

Test accuracy %	Coherent	Incoherent	Constraint
ResNet-18	44.8	**97.7**	56.9

Is prediction the shortest path?
This is the real task!

Are individual edge predictions correct?

Is output a path?

Kareem Ahmed, Eric Wang, Kai-Wei Chang and Guy Van den Broeck. Neuro-Symbolic Entropy Regularization, 2021.

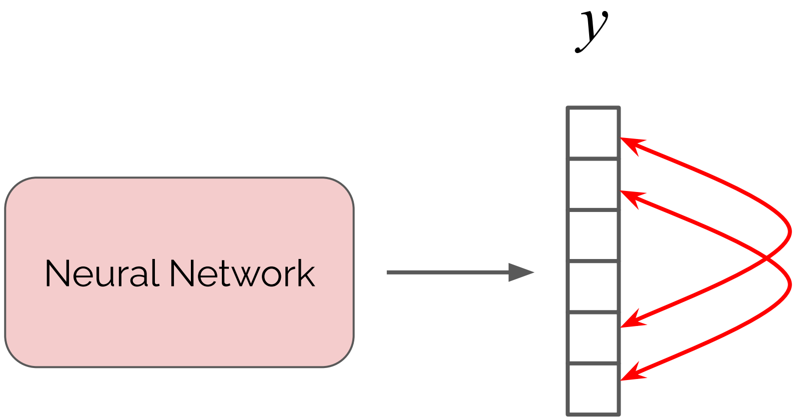
pylon

A PyTorch Framework for Learning with Constraints

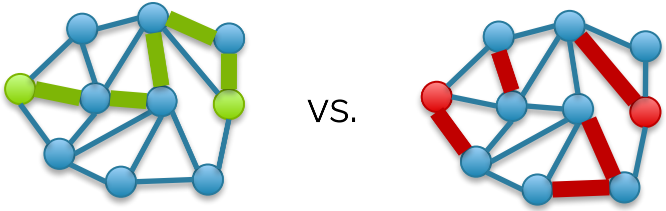
Kareem Ahmed   Tao Li   Thy Ton   Quan Guo,
Kai-Wei Chang   Parisa Kordjamshidi   Vivek Srikumar
Guy Van den Broeck   Sameer Singh

http://pylon-lib.github.io

Declarative Knowledge of the Output



How is the output structured?
Are all possible outputs valid?



How are the outputs related to each other?

Learning this from data is inefficient
Much easier to express this declaratively

pylon

Library that extends PyTorch to allow injection of declarative knowledge

- **Easy to Express Knowledge:** users write arbitrary constraints on the output
- **Integrates with PyTorch:** minimal change to existing code
- **Efficient Training:** compiles into loss that can be efficiently optimized
    - Exact semantic loss (see later)
    - Monte-carlo estimate of loss
    - T-norm approximation
    - *your solver?*

pylon

PyTorch Code

```
for i in range(train_iters):
    ...
    py = model(x)
    ...
    loss = CrossEntropy(py,...)
```

1 Specify knowledge as a predicate

```
def check(y):
    ...
    return isValid
```

pylon

PyTorch Code

```
for i in range(train_iters):
    ...
    py = model(x)
    ...
    loss = CrossEntropy(py,...)

    loss += constraint_loss(check)(py)
```

1 Specify knowledge as a predicate

```
def check(y):
    ...
    return isValid
```

2 Add as loss to training

```
loss += constraint_loss(check)
```

pylon

PyTorch Code

```
for i in range(train_iters):
    ...
    py = model(x)
    ...
    loss = CrossEntropy(py,...)

    loss += constraint_loss(check)(py)
```

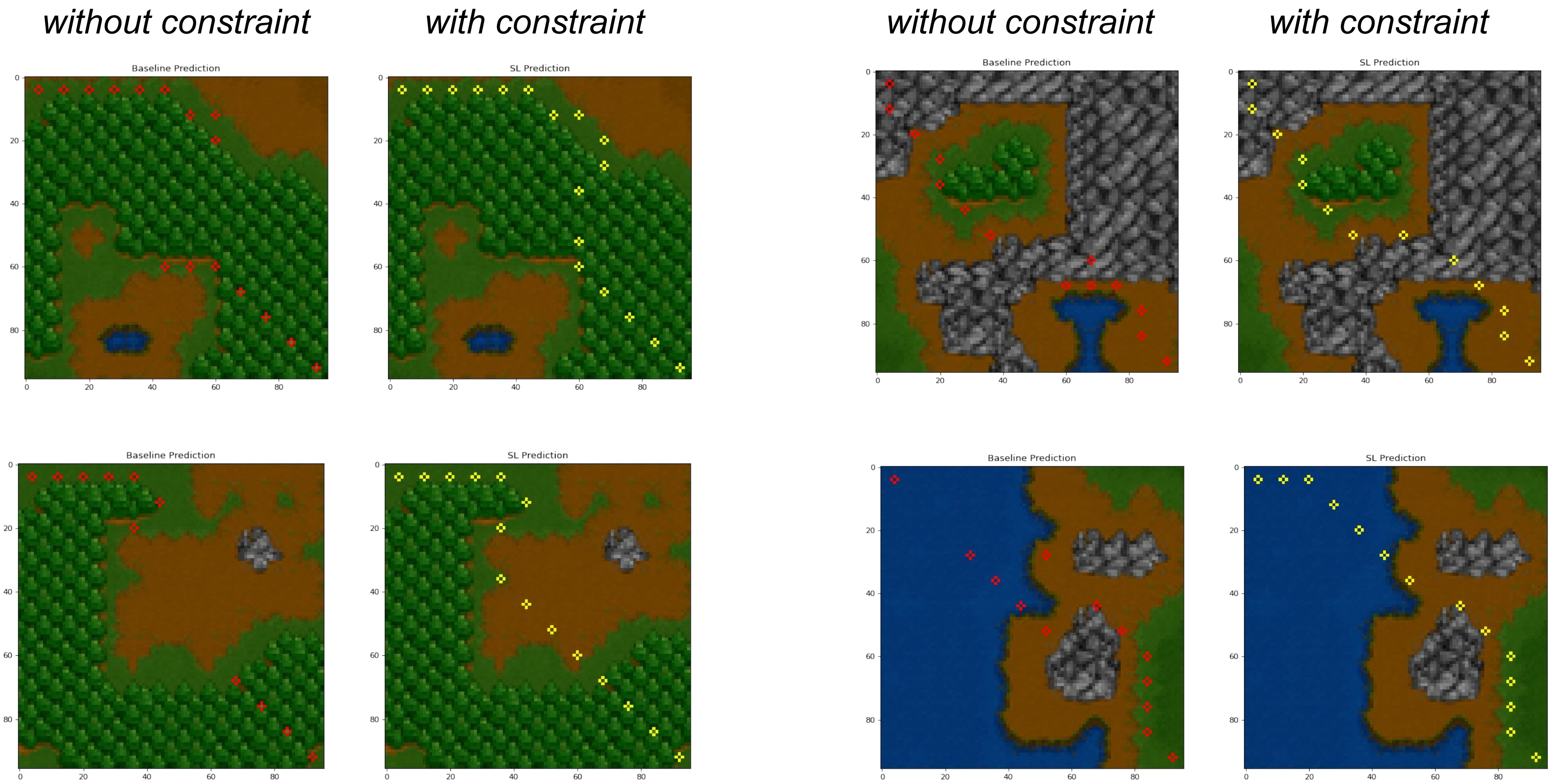
1 Specify knowledge as a predicate

```
def check(y):
    ...
    return isValid
```

2 Add as loss to training

```
loss += constraint_loss(check)
```

3 pylon derives the gradients
(solves a combinatorial problem)


without constraint
with constraint
without constraint
with constraint
Baseline Prediction
SL Prediction
Baseline Prediction
SL Prediction
Baseline Prediction
SL Prediction
Baseline Prediction
SL Prediction

Warcraft min-cost simple-path prediction results

Test accuracy %	Coherent	Incoherent	Constraint
ResNet-18	44.8	**97.7**	56.9
+ Semantic loss	**50.9**	**97.7**	**67.4**

Kareem Ahmed, Eric Wang, Kai-Wei Chang and Guy Van den Broeck. Neuro-Symbolic Entropy Regularization, 2021.

Semantic Loss

*<u>Q</u>: How close is output **p** to satisfying constraint α?*

*<u>A</u>: Semantic loss function L(α,**p**)*

- Axioms, for example:
  - If α constrains to one label, L(α,**p**) is cross-entropy
  - If α implies β then L(α,**p**) ≥ L(β,**p**) (*α more strict*)
- Implied Properties:
  - If α is equivalent to β then L(α,**p**) = L(β,**p**)
  - If **p** is Boolean and satisfies α then L(α,**p**) = 0

SEMANTIC

Loss!

Axioms imply unique semantic loss:

$$\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) \propto -\log \underbrace{\sum_{\mathbf{x} \models \alpha} \prod_{i:\mathbf{x} \models X_i} \mathsf{p}_i \prod_{i:\mathbf{x} \models \neg X_i} (1 - \mathsf{p}_i)}$$

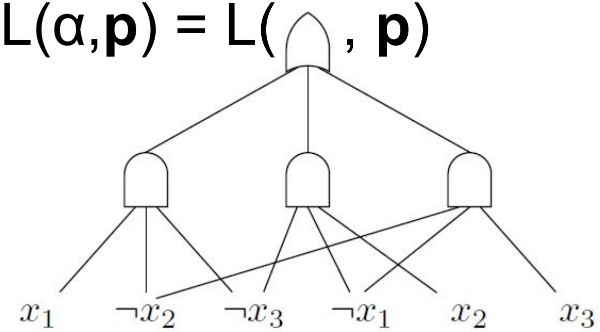
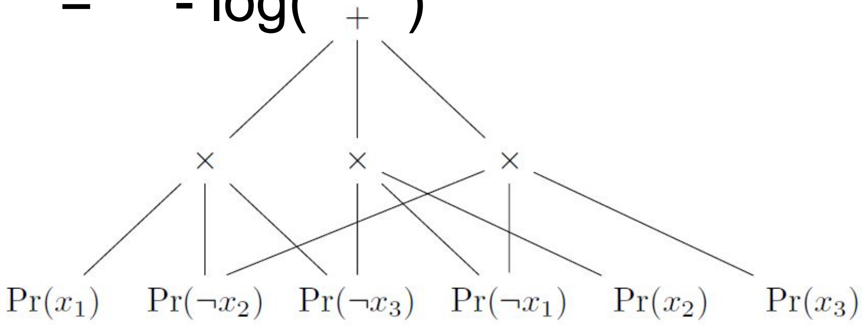
Probability of satisfying constraint α after sampling from neural net output layer **p**

In general: #P-hard ☹️

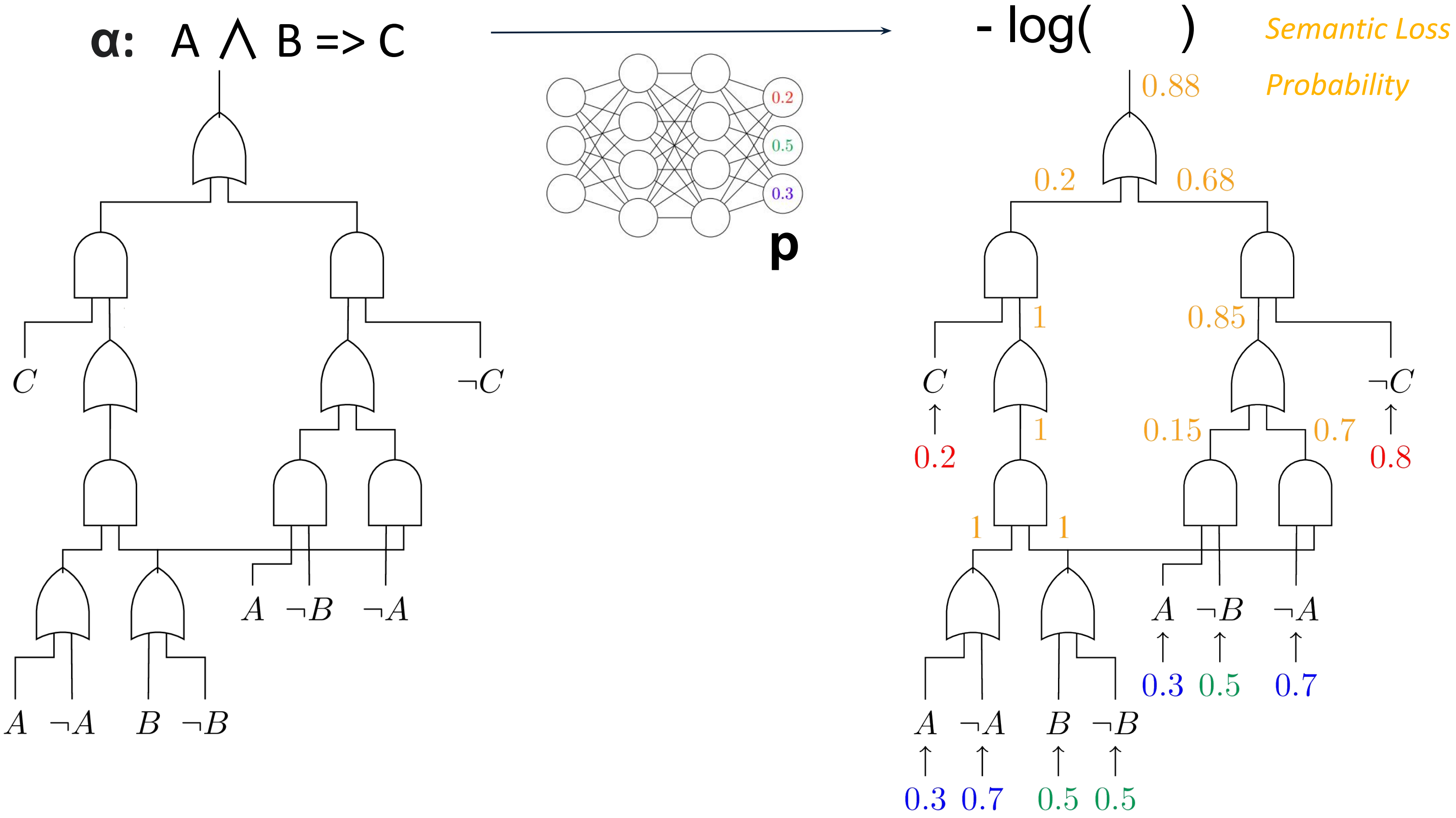
Do this probabilistic-logical reasoning during learning in a computation graph

Circuits = Computation Graphs

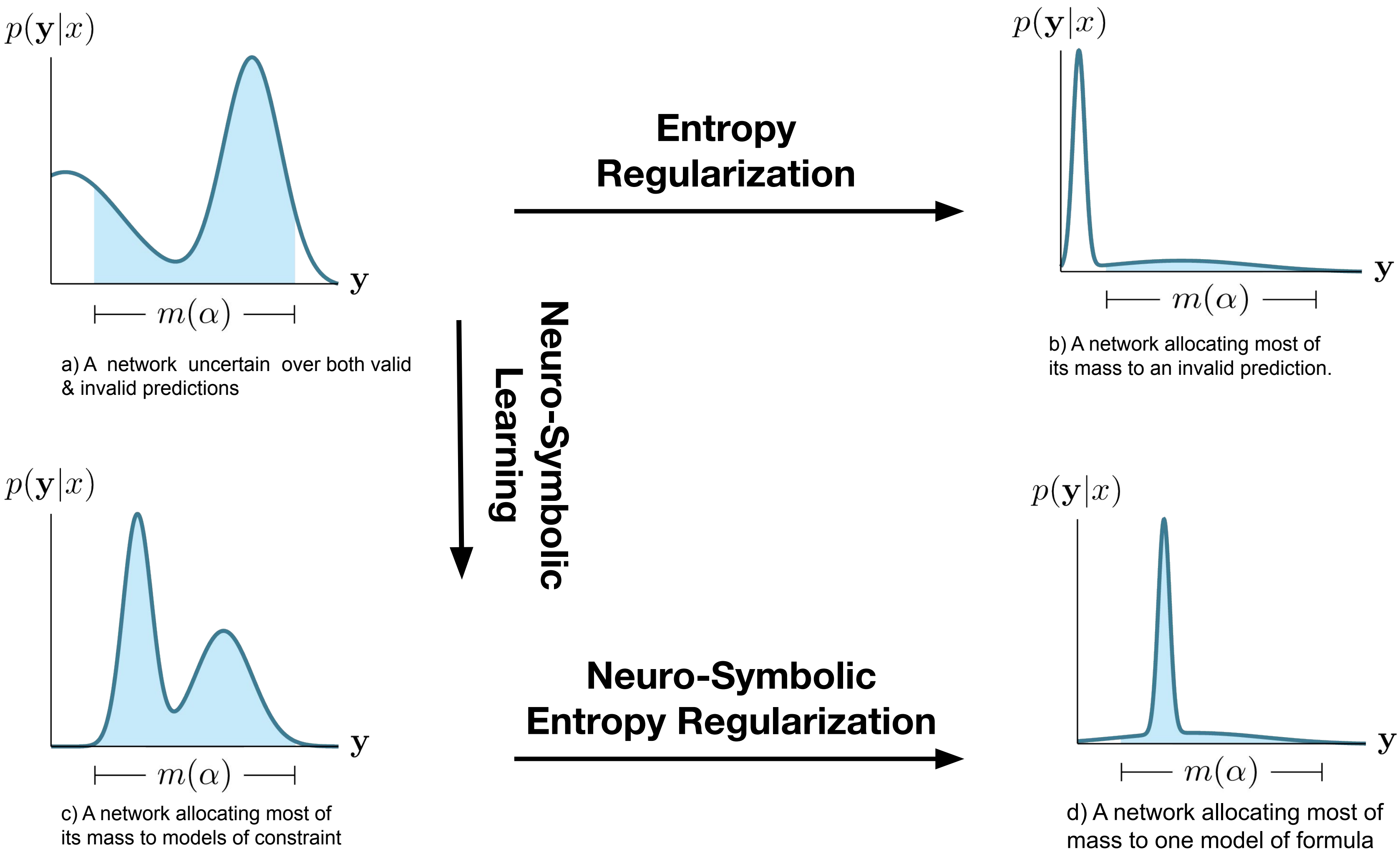
- Logical circuits that can count solutions (#SAT)
  also compute semantic loss efficiently in size of circuit

$L(\alpha,\mathbf{p}) = L($  $, \mathbf{p}) = -\log($  $)$

- Compilation into circuit by SAT solvers (once)
- Add circuit to neural network output in pytorch/tensorflow/...



α: A ∧ B => C
- log( )
Semantic Loss
Probability
p
0.2
0.5
0.3
0.88
0.2
0.68
1
0.85
1
0.15
0.7
1
1
C
¬C
0.2
0.8
A
¬B
¬A
0.3
0.5
0.7
A
¬A
B
¬B
0.3
0.7
0.5
0.5


$p(\mathbf{y}|x)$
$\mathbf{y}$
$m(\alpha)$
a) A network uncertain over both valid & invalid predictions
Entropy Regularization
$p(\mathbf{y}|x)$
$\mathbf{y}$
$m(\alpha)$
b) A network allocating most of its mass to an invalid prediction.
Neuro-Symbolic Learning
$p(\mathbf{y}|x)$
$\mathbf{y}$
$m(\alpha)$
c) A network allocating most of its mass to models of constraint
Neuro-Symbolic Entropy Regularization
$p(\mathbf{y}|x)$
$\mathbf{y}$
$m(\alpha)$
d) A network allocating most of mass to one model of formula

Two complementary neuro-symbolic losses

Neural net $P(\mathbf{Y}|x)$

Constraint α

Semantic Loss

$$P(\alpha|x)\uparrow:\ -\log P(\alpha|x)\downarrow$$

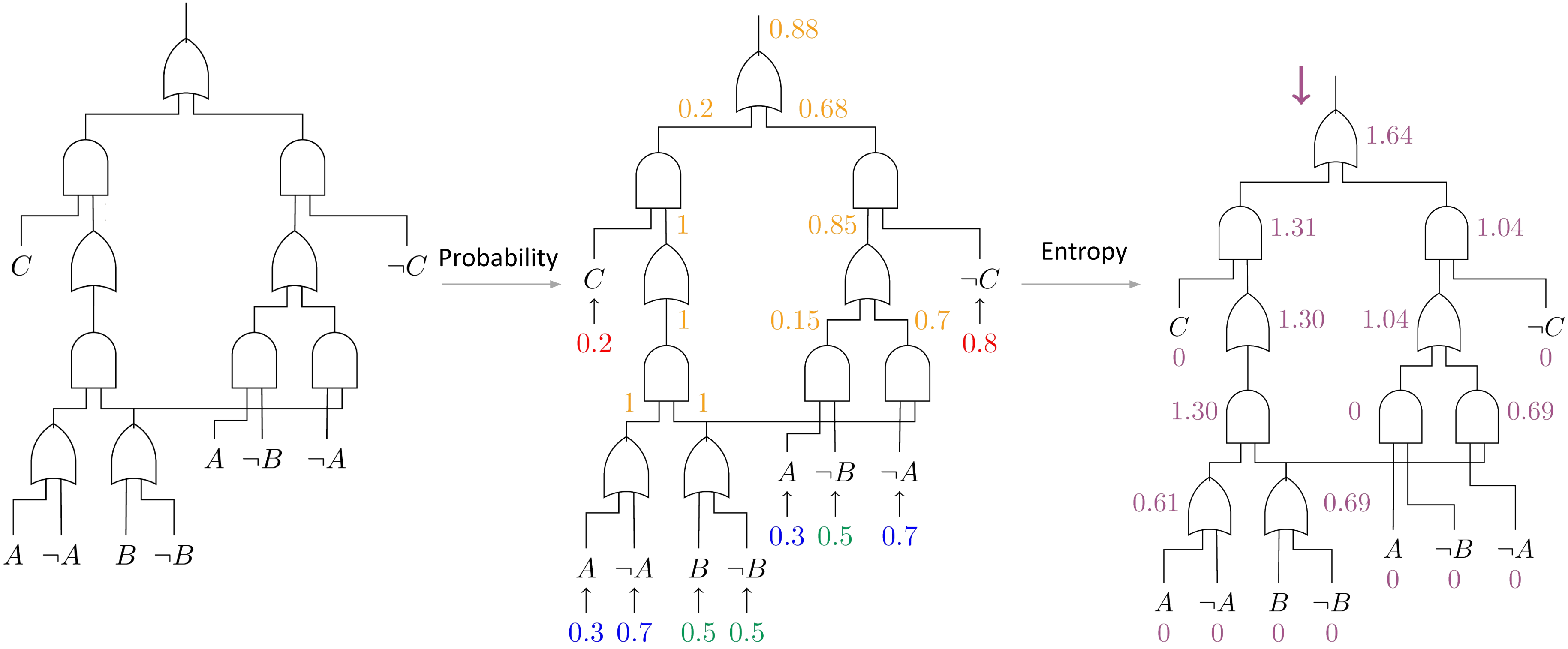
Neuro-symbolic Entropy-regularization

$$-\mathbb{E}_{P(\mathbf{Y}|x,\alpha)}[\log P(\mathbf{Y}|x,\alpha)]$$

Warcraft min-cost simple-path prediction results

Test accuracy %	Coherent	Incoherent	Constraint
ResNet-18	44.8	97.7	56.9
Semantic loss	50.9	97.7	67.4
+ Full Entropy	51.5	97.6	67.7
+ NeSy Entropy	**55.0**	**97.9**	**69.8**

Kareem Ahmed, Eric Wang, Kai-Wei Chang and Guy Van den Broeck. Neuro-Symbolic Entropy Regularization, 2021.



C
¬C
A ¬A B ¬B
A ¬B ¬A
Probability
0.88
0.2 0.68
1 0.85
1 0.15 0.7
0.2 0.8
1 1
0.3 0.7 0.5 0.5
0.3 0.5 0.7
Entropy
1.64
1.31 1.04
1.30 1.04
1.30 0 0.69
0.61 0.69
0 0 0 0 0 0 0 0 0

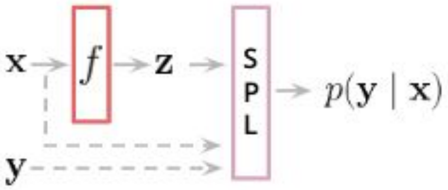
Joint entity-relation extraction in natural language processing

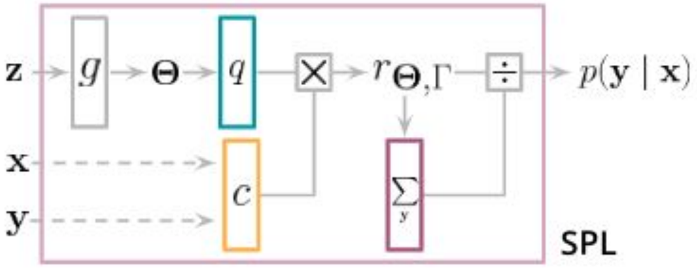
#		3	5	10	15	25	50	75
ACE05	Baseline	4.92 ± 1.12	7.24 ± 1.75	13.66 ± 0.18	15.07 ± 1.79	21.65 ± 3.41	28.96 ± 0.98	33.02 ± 1.17
	Self-training	7.72 ± 1.21	12.83 ± 2.97	16.22 ± 3.08	17.55 ± 1.41	27.00 ± 3.66	32.90 ± 1.71	37.15 ± 1.42
	Product t-norm	8.89 ± 5.09	14.52 ± 2.13	19.22 ± 5.81	21.80 ± 7.67	30.15 ± 1.01	34.12 ± 2.75	37.35 ± 2.53
	Semantic Loss	12.00 ± 3.81	14.92 ± 3.14	22.23 ± 3.64	27.35 ± 3.10	30.78 ± 0.68	36.76 ± 1.40	38.49 ± 1.74
	+ Full Entropy	**14.80** ± 3.70	15.78 ± 1.90	23.34 ± 4.07	28.09 ± 1.46	31.13 ± 2.26	36.05 ± 1.00	39.39 ± 1.21
	+ NeSy Entropy	14.72 ± 1.57	**18.38** ± 2.50	**26.41** ± 0.49	**31.17** ± 1.68	**35.85** ± 0.75	**37.62** ± 2.17	**41.28** ± 0.46
SciERC	Baseline	2.71 ± 1.10	2.94 ± 1.00	3.49 ± 1.80	3.56 ± 1.10	8.83 ± 1.00	12.32 ± 3.00	12.49 ± 2.60
	Self-training	3.56 ± 1.40	3.04 ± 0.90	4.14 ± 2.60	3.73 ± 1.10	9.44 ± 3.80	14.82 ± 1.20	13.79 ± 3.90
	Product t-norm	**6.50** ± 2.00	8.86 ± 1.20	10.92 ± 1.60	13.38 ± 0.70	13.83 ± 2.90	19.20 ± 1.70	19.54 ± 1.70
	Semantic Loss	6.47 ± 1.02	**9.31** ± 0.76	11.50 ± 1.53	12.97 ± 2.86	14.07 ± 2.33	20.47 ± 2.50	23.72 ± 0.38
	+ Full Entropy	6.26 ± 1.21	8.49 ± 0.85	11.12 ± 1.22	14.10 ± 2.79	17.25 ± 2.75	**22.42** ± 0.43	24.37 ± 1.62
	+ NeSy Entropy	6.19 ± 2.40	8.11 ± 3.66	**13.17** ± 1.08	**15.47** ± 2.19	**17.45** ± 1.52	22.14 ± 1.46	**25.11** ± 1.03

Kareem Ahmed, Eric Wang, Kai-Wei Chang and Guy Van den Broeck. Neuro-Symbolic Entropy Regularization, 2022.

Semantic Probabilistic Layers

- How to give a 100% guarantee that Boolean constraints will be satisfied?
- Bake the constraint into the neural network as a special layer





- Secret sauce is again tractable circuits – computation graphs for reasoning

Kareem Ahmed, Stefano Teso, Kai-Wei Chang, Guy Van den Broeck and Antonio Vergari. Semantic Probabilistic Layers for Neuro-Symbolic Learning, 2022.

Warcraft Shortest Path

GROUND TRUTH | RESNET-18 | SEMANTIC LOSS | SPL (ours)

Hierarchical Multi-Label Classification

“if the image is classified as a dog, it must also be classified as an animal”

“if the image is classified as an animal, it must be classified as either cat or dog”

Dataset	Exact Match	
	HMCNN	MLP+SPL
CellCycle	3.05 ± 0.11	$\mathbf{3.79 \pm 0.18}$
Derisi	1.39 ± 0.47	$\mathbf{2.28 \pm 0.23}$
Eisen	5.40 ± 0.15	$\mathbf{6.18 \pm 0.33}$
Expr	4.20 ± 0.21	$\mathbf{5.54 \pm 0.36}$
Gasch1	3.48 ± 0.96	$\mathbf{4.65 \pm 0.30}$
Gasch2	3.11 ± 0.08	$\mathbf{3.95 \pm 0.28}$
Seq	5.24 ± 0.27	$\mathbf{7.98 \pm 0.28}$
Spo	$\mathbf{1.97 \pm 0.06}$	1.92 ± 0.11
Diatoms	48.21 ± 0.57	$\mathbf{58.71 \pm 0.68}$
Enron	5.97 ± 0.56	$\mathbf{8.18 \pm 0.68}$
Imclef07a	79.75 ± 0.38	$\mathbf{86.08 \pm 0.45}$
Imclef07d	76.47 ± 0.35	$\mathbf{81.06 \pm 0.68}$

Neuro-Symbolic Learning Settings

Learn

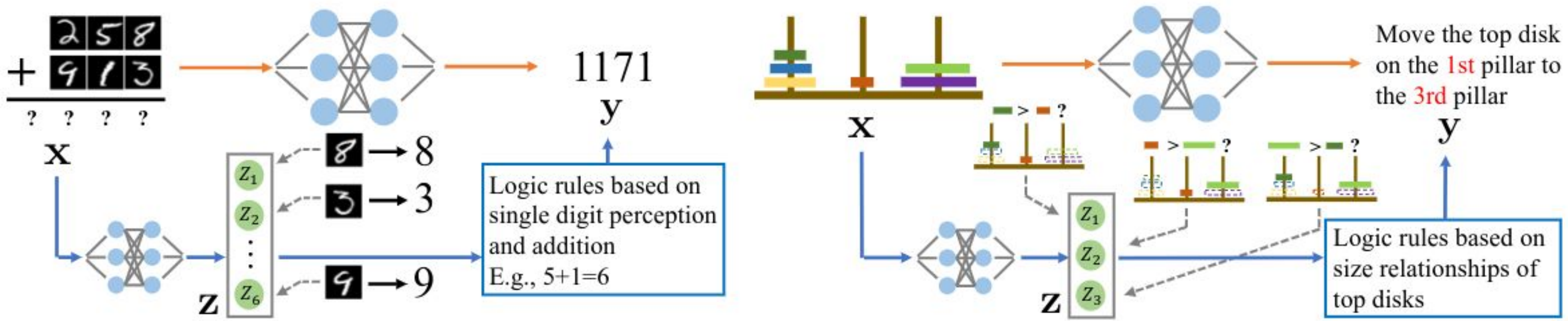
1. neural network given symbols and constraints and data

2. neural network and constraints given symbols and data

3. neural network and constraints and symbols given data

Everyone is working on 1. Ongoing work on 2.

Neuro-Symbolic Joint Training



Learn invariant features using neural networks. Learn logic to tie it all together.

Ask Yitao Liang, Anji Liu

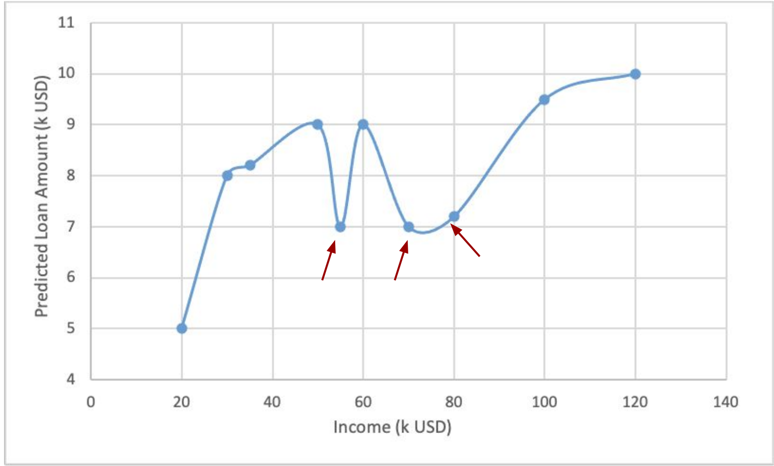
Neuro-Symbolic Joint Training

Model	Multi-digit addition [test seq length + train/test img]						Tower of Hanoi		
	5 w/ test	10 w/ test	20 w/ test	5 w/ train	10 w/ train	20 w/ train	Task #1	Task #2	Task #3
DeepProbLog†	88.30	77.46	timeout	94.92	89.74	timeout	89.28	97.96	89.33
LSTM	81.40	56.97	39.05	88.92	77.40	63.23	78.26	**98.32**	74.36
DNC	81.49	59.64	33.83	81.88	59.96	37.85	76.20	97.87	73.87
NToC(ours)	**89.82**	**77.97**	**63.55**	**89.97**	**86.07**	**71.96**	**85.16**	97.94	**85.49**

Learn invariant features using neural networks. Learn logic to tie it all together.

Ask Yitao Liang, Anji Liu

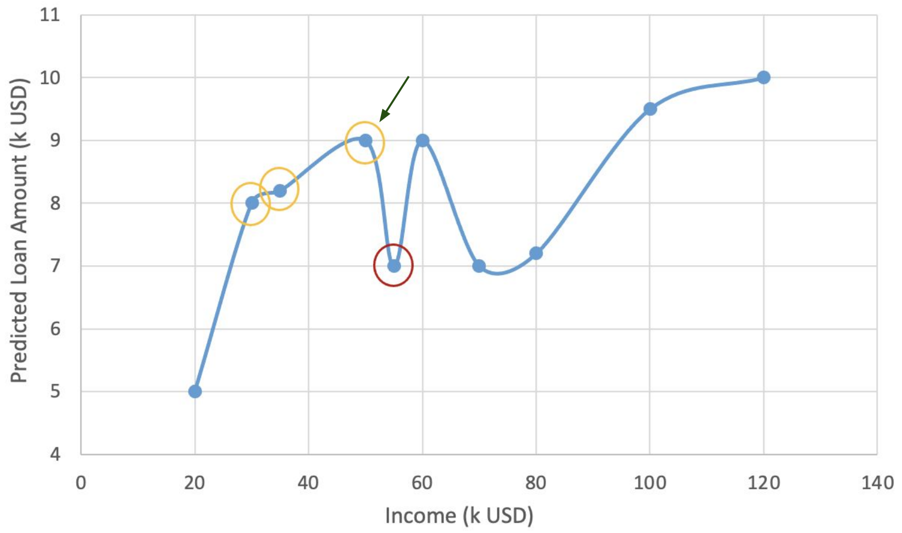
Predict Loan Amount



Neural Network Model: Increasing income can decrease the approved loan amount

Monotonicity (Prior Knowledge):
Increasing income should increase the approved loan amount

Counterexamples



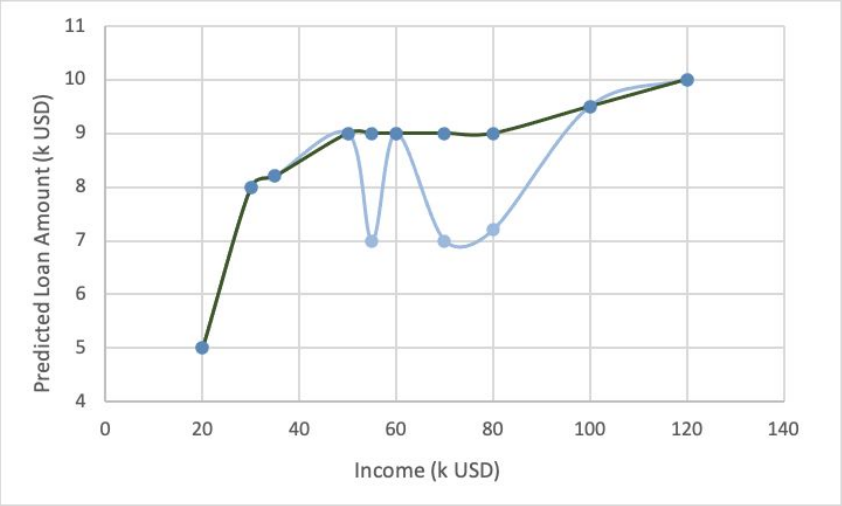
$$\exists x, y \; x \leq y \implies f(x) > f(y)$$

Computed using SMT(LRA) logical reasoning solver

Maximal counterexamples (largest violation) using OMT

Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein and Guy Van den Broeck. Counterexample-Guided Learning of Monotonic Neural Networks, *NeurIPS*, 2020.

Counterexample-Guided Predictions



Monotonic Envelope:

- Replace each prediction by its maximal counterexample
- Envelope construction is online (during prediction)
- Guarantees monotonic predictions for any ReLU neural net

- Works for high-dimensional input
- Works for multiple monotonic features

Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein and Guy Van den Broeck. Counterexample-Guided Learning of Monotonic Neural Networks, *NeurIPS*, 2020.

Monotonic Envelope: Performance

Dataset	Feature	NN_b	Envelope
Auto-MPG	Weight	9.33±3.22	**9.19±3.41**
	Displ.	9.33±3.22	9.63±2.61
	W,D	9.33±3.22	9.63±2.61
	W,D,HP	9.33±3.22	9.63±2.61
Boston	Rooms	14.37±2.4	**14.19±2.28**
	Crime	14.37±2.4	**14.02±2.17**

Dataset	Feature	NN_b	Envelope
Heart	Trestbps	0.85±0.04	0.85±0.04
	Chol.	0.85±0.04	0.85±0.05
	T,C	0.85±0.04	0.85±0.05
Adult	Cap. Gain	0.84	0.84
	Hours	0.84	0.84

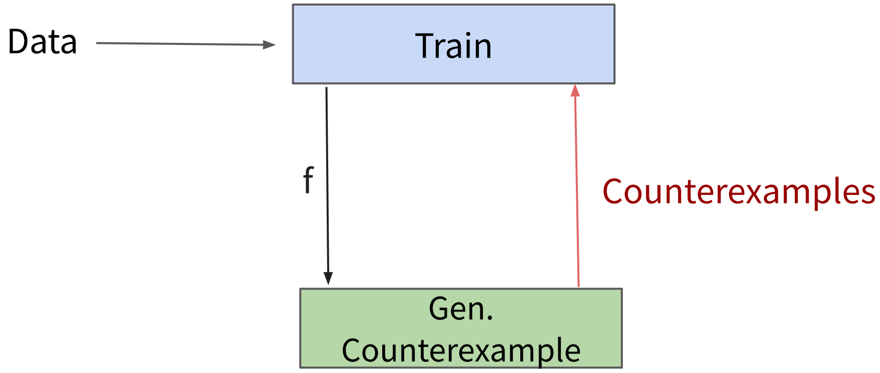
Guaranteed monotonicity at little to no cost

Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein and Guy Van den Broeck. Counterexample-Guided Learning of Monotonic Neural Networks, *NeurIPS*, 2020.

Counterexample-Guided Learning

How to use monotonicity to improve model quality?

“Monotonicity as inductive bias”



T Epochs

Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein and Guy Van den Broeck. Counterexample-Guided Learning of Monotonic Neural Networks, *NeurIPS*, 2020.

Counterexample-Guided Learning: Performance

Dataset	Feature	NN_b	CGL
Auto-MPG	Weight	9.33±3.22	**9.04±2.76**
	Displ.	9.33±3.22	**9.08±2.87**
	W,D	9.33±3.22	**8.86±2.67**
	W,D,HP	9.33±3.22	**8.63±2.21**
Boston	Rooms	14.37±2.4	**12.24±2.87**
	Crime	14.37±2.4	**11.66±2.89**

Dataset	Feature	NN_b	CGL
Heart	Trestbps	0.85±0.04	**0.86±0.02**
	Chol.	**0.85±0.04**	**0.85±0.05**
	T,C	0.85±0.04	**0.86±0.06**
Adult	Cap. Gain	**0.84**	**0.84**
	Hours	**0.84**	**0.84**

Monotonicity is a *great* inductive bias for learning

Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein and Guy Van den Broeck. Counterexample-Guided Learning of Monotonic Neural Networks, *NeurIPS*, 2020.

COMET:
Counterexample-Guided Monotonicity Enforced Training

Table 4: Monotonicity is an effective inductive bias. COMET outperforms Min-Max networks on all datasets. COMET outperforms DLN in regression datasets and achieves similar results in classification datasets.

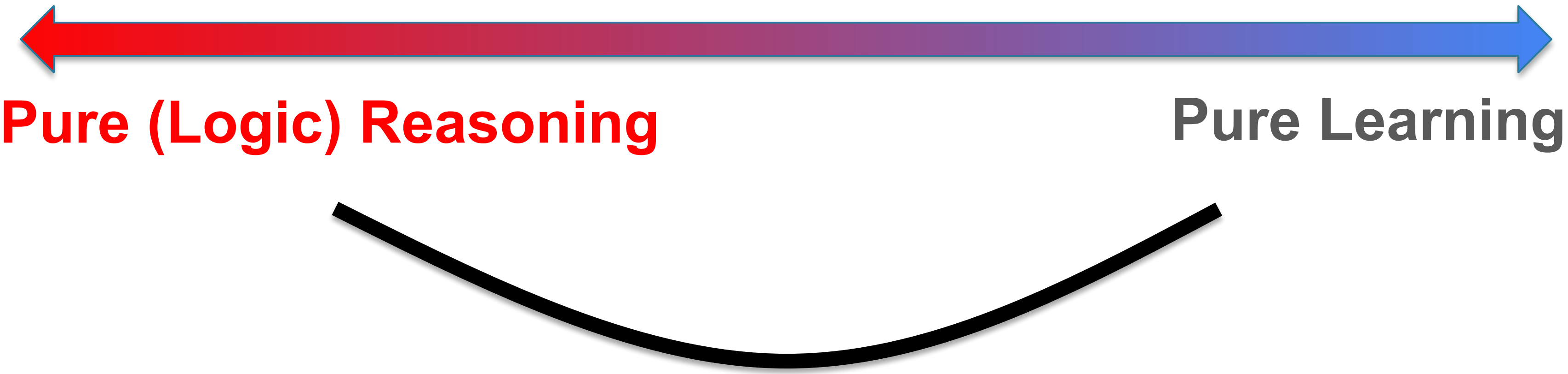
Dataset	**Features**	**Min-Max**	**DLN**	COMET
Auto-MPG	Weight	9.91±1.20	16.77±2.57	**8.92±2.93**
	Displ.	11.78±2.20	16.67±2.25	**9.11±2.25**
	W,D	11.60±0.54	16.56±2.27	**8.89±2.29**
	W,D,HP	10.14±1.54	13.34±2.42	**8.81±1.81**
Boston	Rooms	30.88±13.78	15.93±1.40	**11.54±2.55**
	Crime	25.89±2.47	12.06±1.44	**11.07±2.99**

Dataset	**Features**	**Min-Max**	**DLN**	COMET
Heart	Trestbps	0.75±0.04	0.85±0.02	**0.86±0.03**
	Chol.	0.75±0.04	0.85±0.04	**0.87±0.03**
	T,C	0.75±0.04	**0.86±0.02**	**0.86±0.03**
Adult	Cap. Gain	0.77	**0.84**	**0.84**
	Hours	0.73	**0.85**	0.84

COMET = Provable Guarantees + SotA Results

Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein and Guy Van den Broeck. Counterexample-Guided Learning of Monotonic Neural Networks, *NeurIPS*, 2020.

The AI Dilemma



- Knowledge is (hidden) everywhere in ML
- A little bit of reasoning goes a long way!

Deep learning with structured output constraints
Learning monotonic neural networks

Outline

1. The paradox of learning to reason from data

   ~~deep learning~~

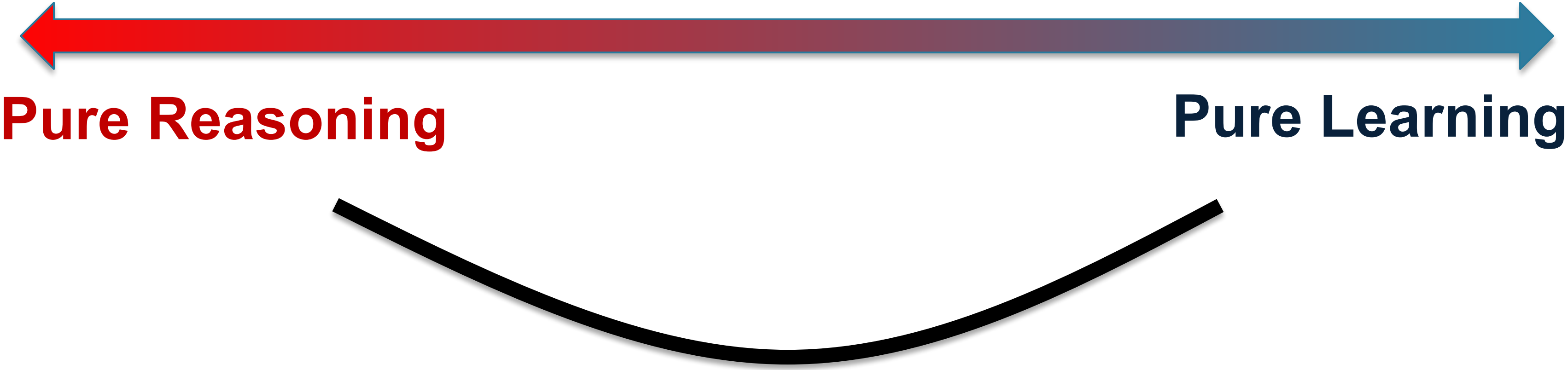
2. Tractable deep generative models

   probabilistic reasoning + *deep learning*

3. Learning with symbolic knowledge

   logical reasoning + *deep learning*

The AI Dilemma



Integrate reasoning into modern deep learning algorithms

Thanks

This was the work of many wonderful students/postdocs/collaborators!

References: http://starai.cs.ucla.edu/publications/