UCLA Computer Science

STAR AI RESEARCH LAB UCLA

Computational Probabilistic Models

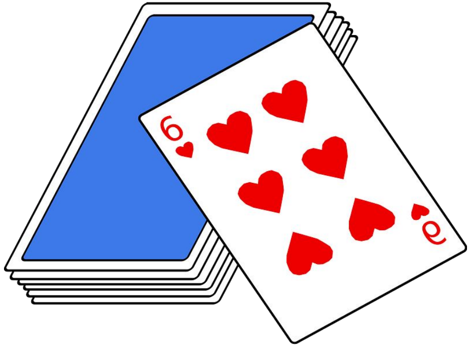
Guy Van den Broeck

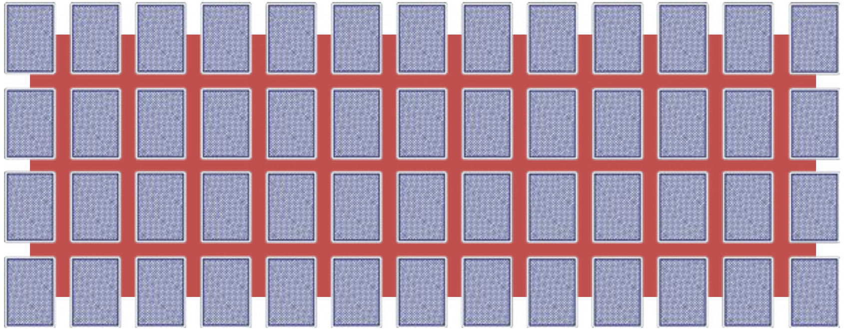
Perspectives in AI - Mar 31, 2022

What is the right abstraction for distributions?

Probabilistic graphical models is how we do probabilistic AI!

Graphical models of variable-level (in)dependence are a broken abstraction.





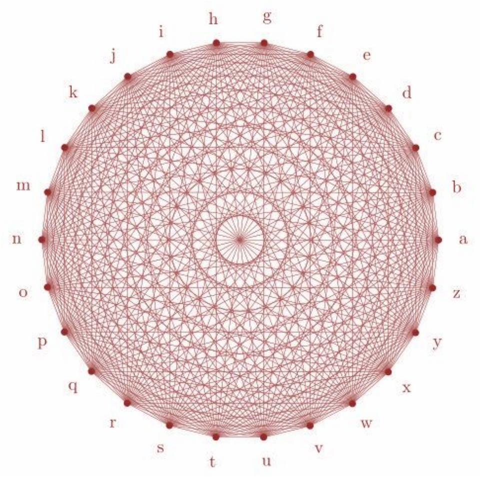
[VdB KRR15]

What is the right abstraction for distributions?

Probabilistic graphical models is how we do probabilistic AI!

Graphical models of variable-level (in)dependence are a broken abstraction.

3.14 Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)



[VdB KRR15]

What is the right abstraction for distributions?

Probabilistic graphical models is how we do probabilistic AI!

Graphical models of variable-level (in)dependence are a broken abstraction.

Bean Machine

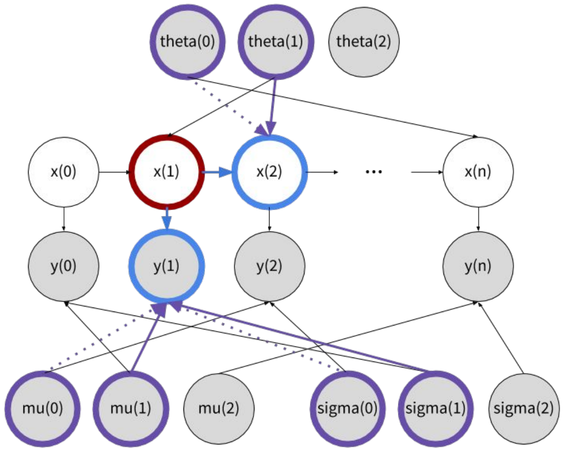
$$\mu_k \sim \text{Normal}(\alpha, \beta)$$

$$\sigma_k \sim \text{Gamma}(\nu, \rho)$$

$$\theta_k \sim \text{Dirichlet}(\kappa)$$

$$x_i \sim \begin{cases} \text{Categorical}(init) & \text{if } i = 0 \\ \text{Categorical}(\theta_{x_{i-1}}) & \text{if } i > 0 \end{cases}$$

$$y_i \sim \text{Normal}(\mu_{x_i}, \sigma_{x_i})$$



[Tehrani et al. PGM20]

Computational Abstractions

Let us think of probability as something that is computed.

Abstraction = Structure of Computation

Two levels of abstraction:

Probabilistic Programs	“High-level code”
Probabilistic Circuits	“Machine code”

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*

Intractable and *tractable* models

Fully factorized

NaiveBayes *AndOrGraphs* *PDGs*

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

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

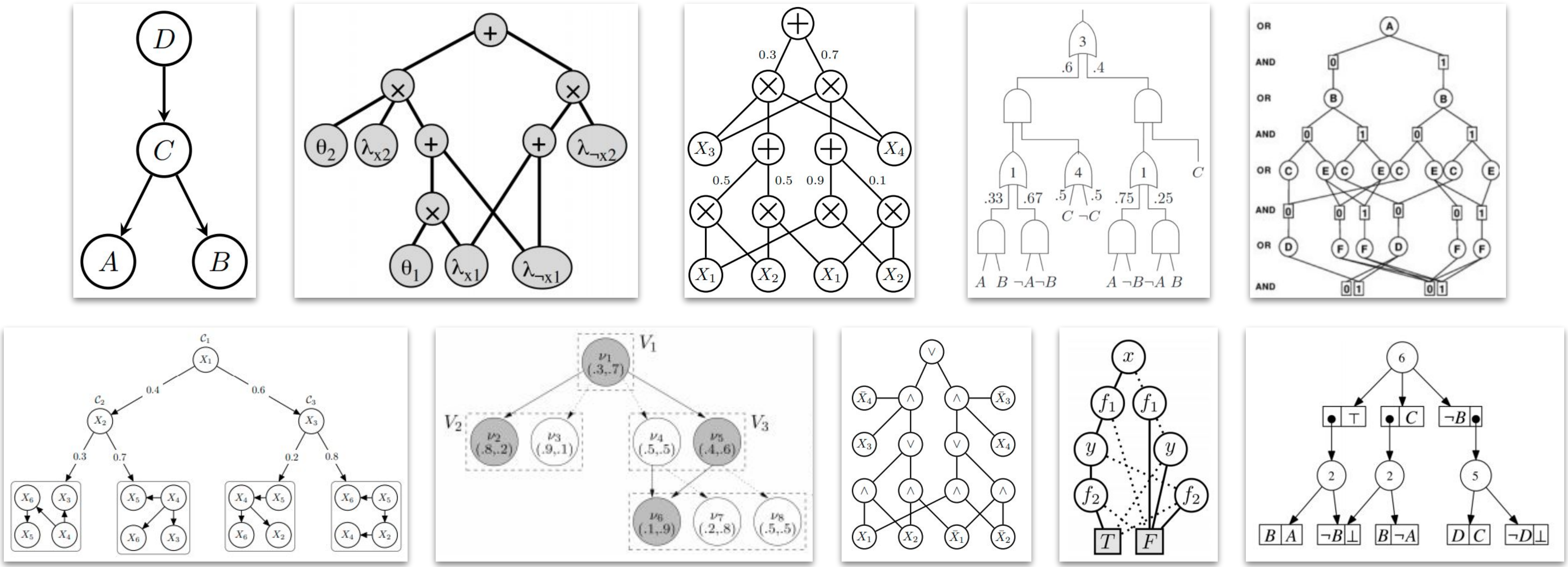
DPPs *FVSBNs* *TACs* *IAFs* *NAFs* *RAEs*

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

RealNVP *MNs*

a ***unifying framework*** for tractable models

Tractable Probabilistic Models

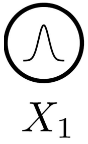


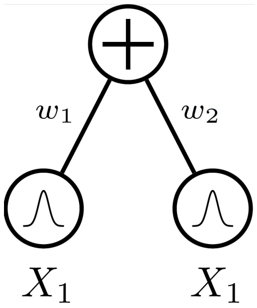
"*Every talk needs a joke and a literature overview slide, not necessarily distinct*"

- after Ron Graham

Probabilistic circuits

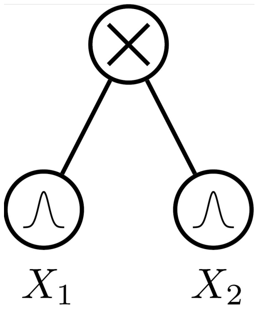
computational graphs that recursively define distributions





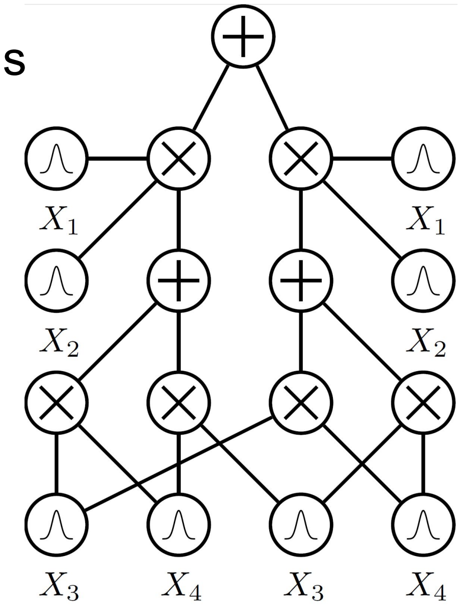
$p(X_1) = w_1 p_1(X_1) + w_2 p_2(X_1)$

⇒ *mixtures*



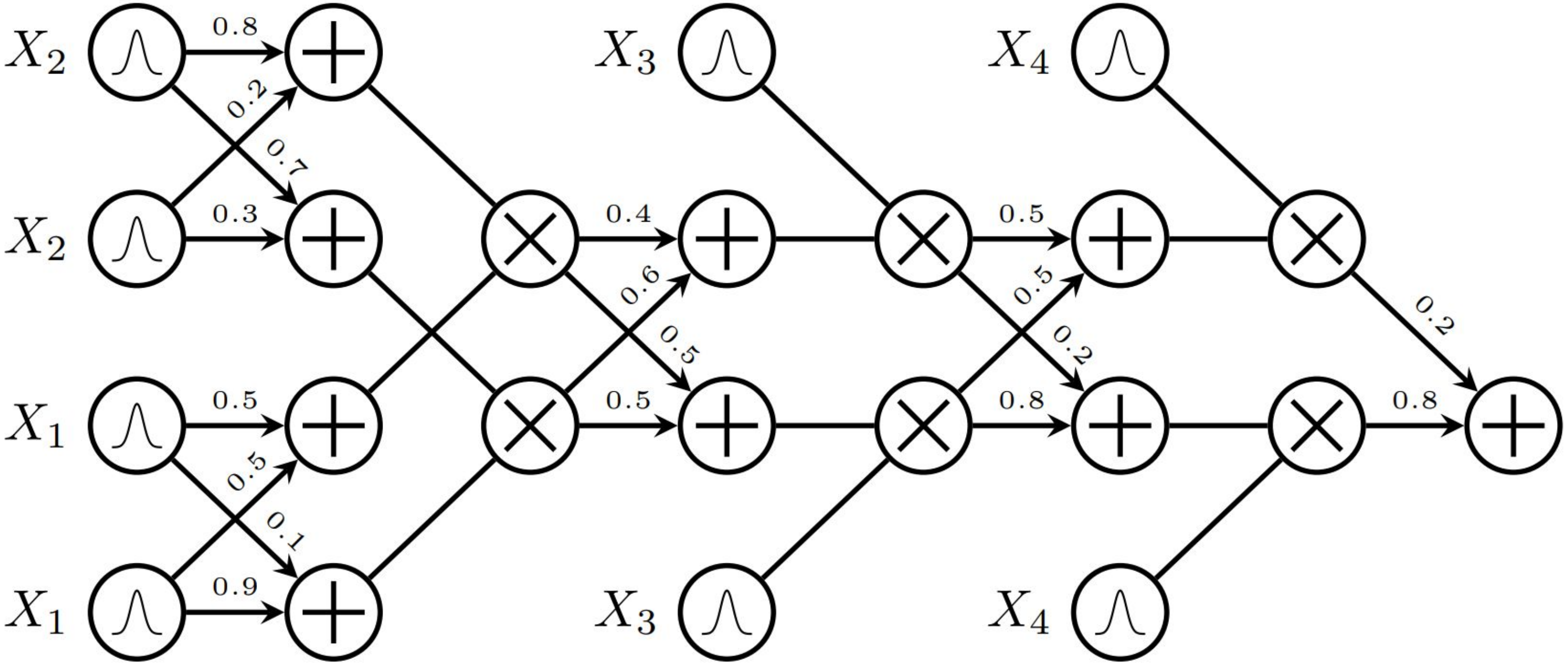
$p(X_1, X_2) = p(X_1) \cdot p(X_2)$

⇒ *factorizations*



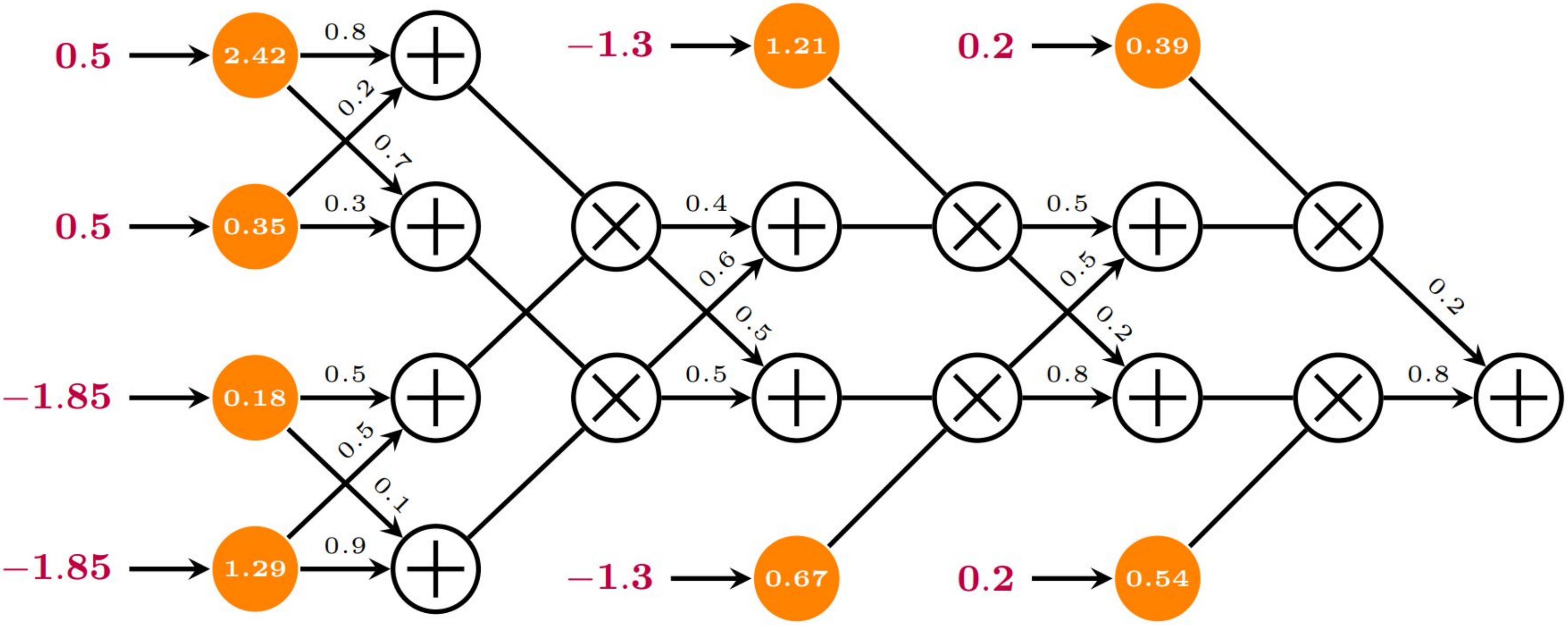
Likelihood

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



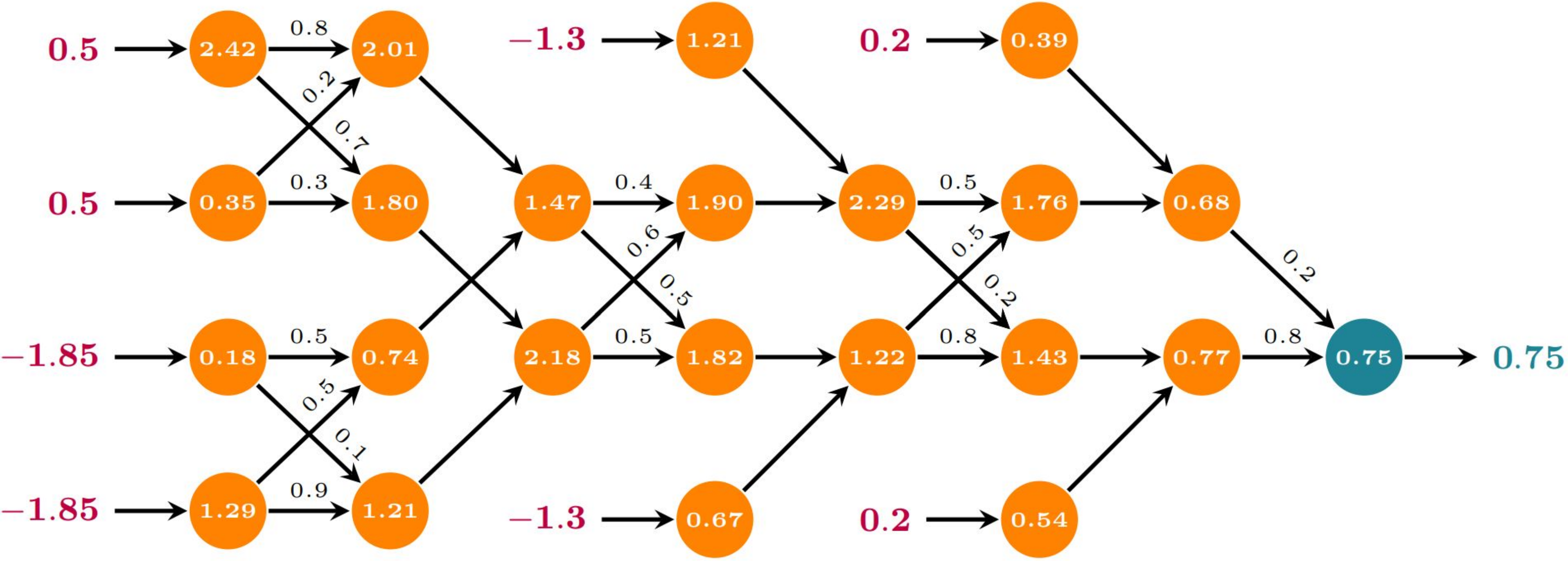
Likelihood

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



Likelihood

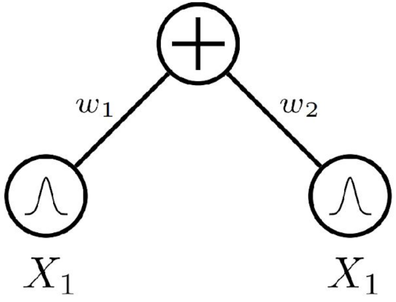
$$p(X_1 = \mathbf{-1.85}, X_2 = \mathbf{0.5}, X_3 = \mathbf{-1.3}, X_4 = \mathbf{0.2})$$



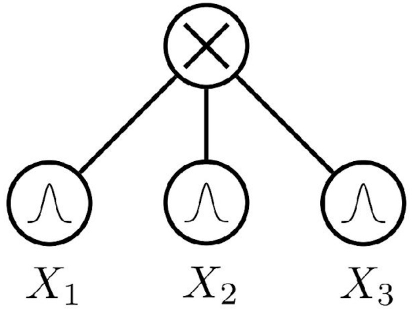
Tractable marginals

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

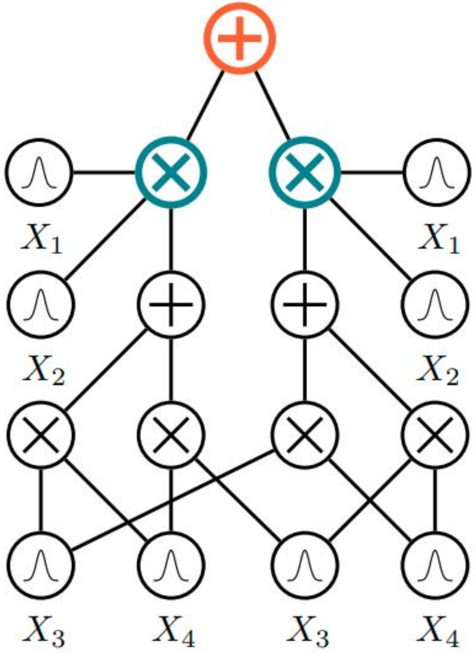
Smoothness + *decomposability* = *tractable MAR*

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

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

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

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



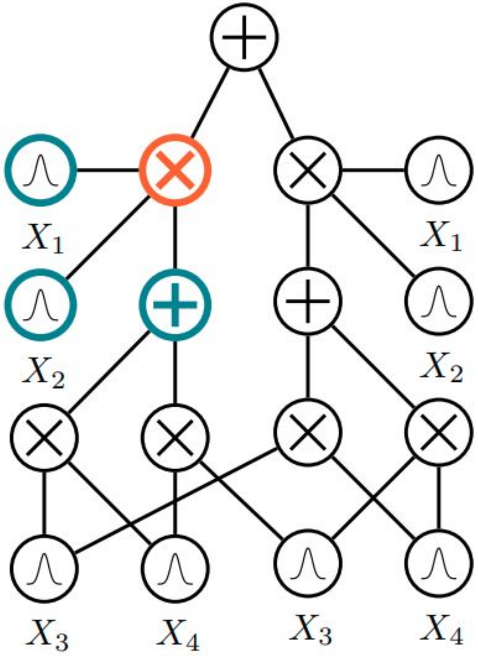
[Darwiche & Marquis JAIR 2001, Poon & Domingos UAI11]

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

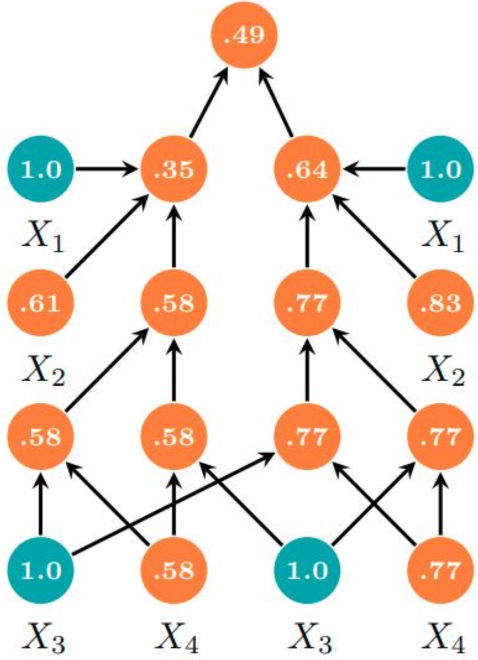
$\Rightarrow$ *linear in circuit size!*

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

- leafs over X_1 and X_3 output $\boldsymbol{Z}_i = \int p(x_i)dx_i$

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

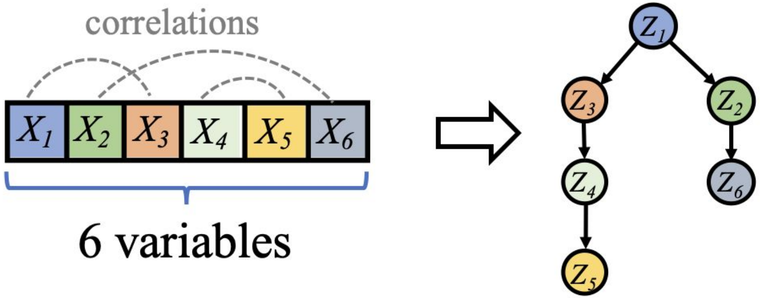
- leafs over X_2 and X_4 output *EVI*
- feedforward evaluation (bottom-up)



Learning Expressive Probabilistic Circuits

Hidden Chow-Liu Trees

Learned **CLT structure** captures strong pairwise dependencies



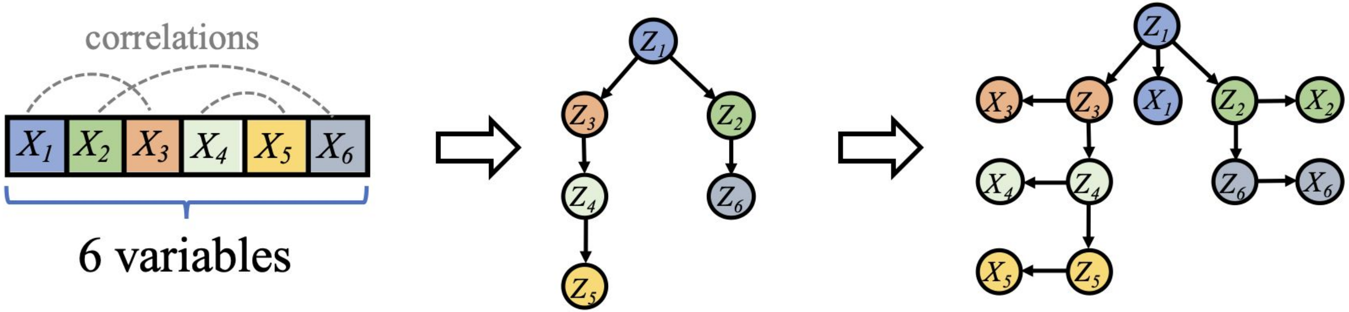
Anji Liu and Guy Van den Broeck. Tractable Regularization of Probabilistic Circuits, *NeurIPS*, 2021.

Learning Expressive Probabilistic Circuits

Hidden Chow-Liu Trees

Learned **CLT structure** captures strong pairwise dependencies

Learned **HCLT structure**

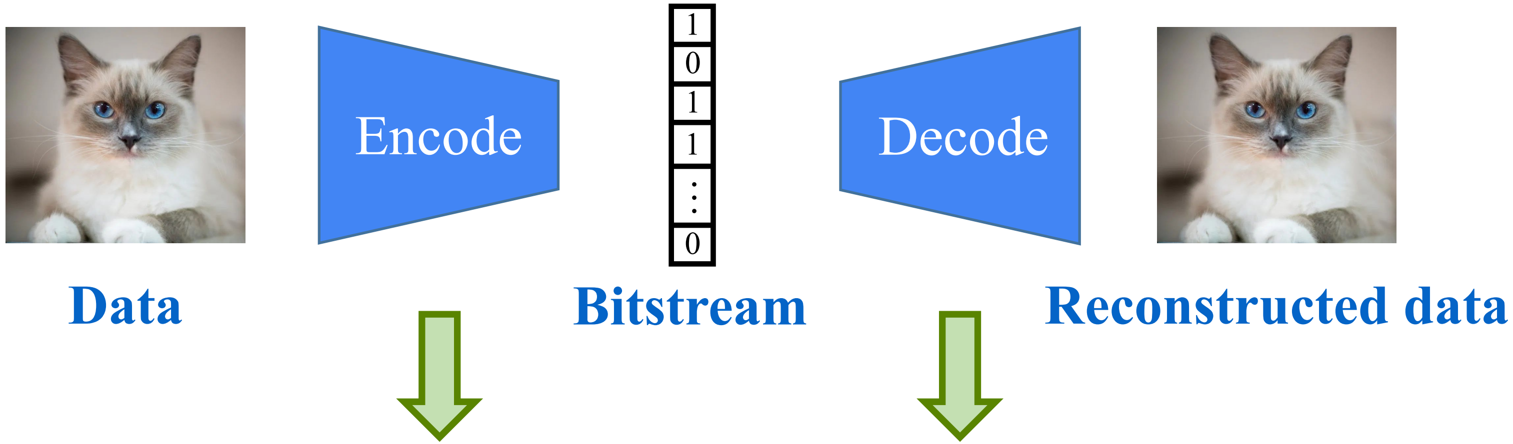


⇨ **Compile** into an equivalent PC

⇨ Mini-batch Stochastic **Expectation Maximization**

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

Lossless Data Compression



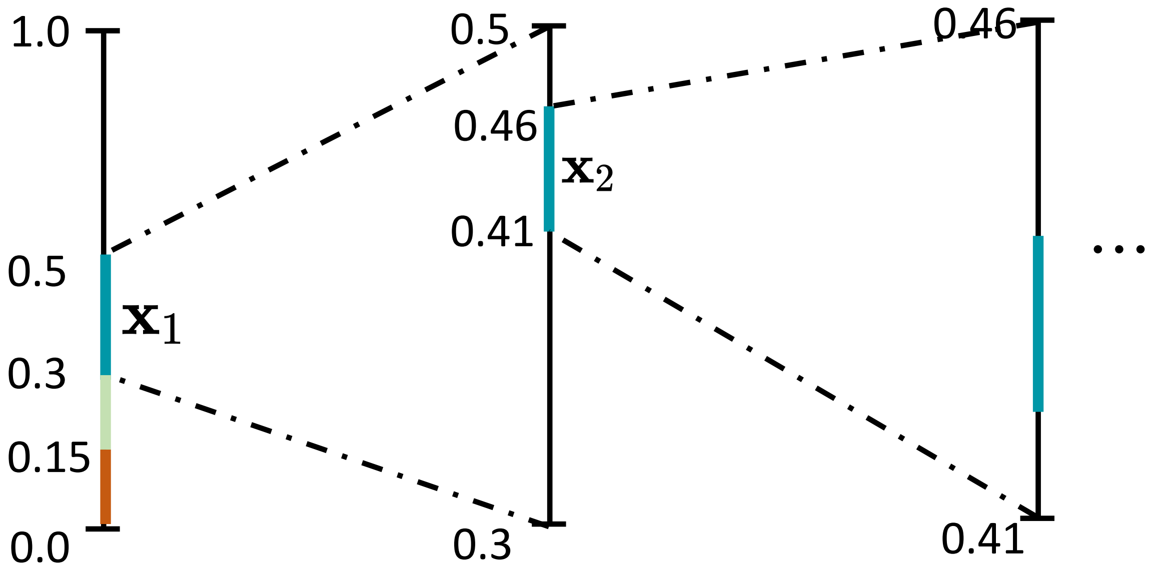
Expressive probabilistic model $p(\boldsymbol{x})$ ⇨ Determines the theoretical limit of compression rate

+

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

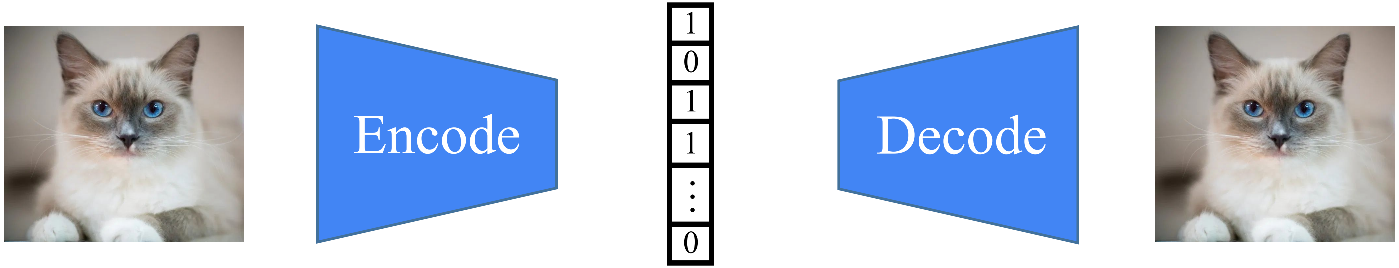
Compress $\mathbf{x}_3$ with $-\log p(\mathbf{x}_3|\mathbf{x}_1, \mathbf{x}_2)$ bits

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

Data **Bitstream** **Reconstructed data**



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.

Fully factorized

NaiveBayes *AndOrGraphs* *PDGs*

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

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

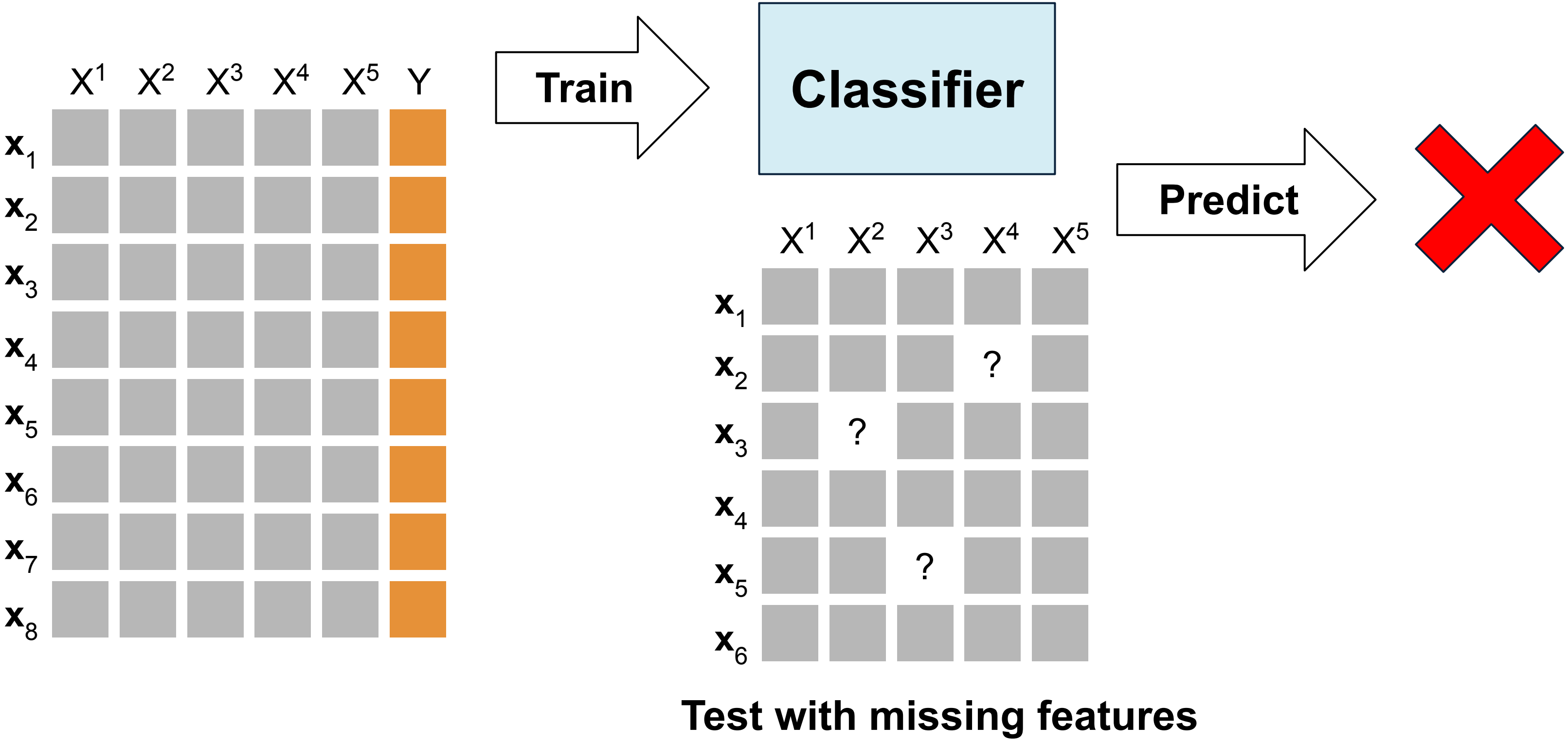
DPPs *FVSBNs* *TACs* *IAFs* *NAFs* *RAEs*

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

RealNVP *MNs*

Expressive models without *compromises*

Prediction with Missing Features



Expected Predictions

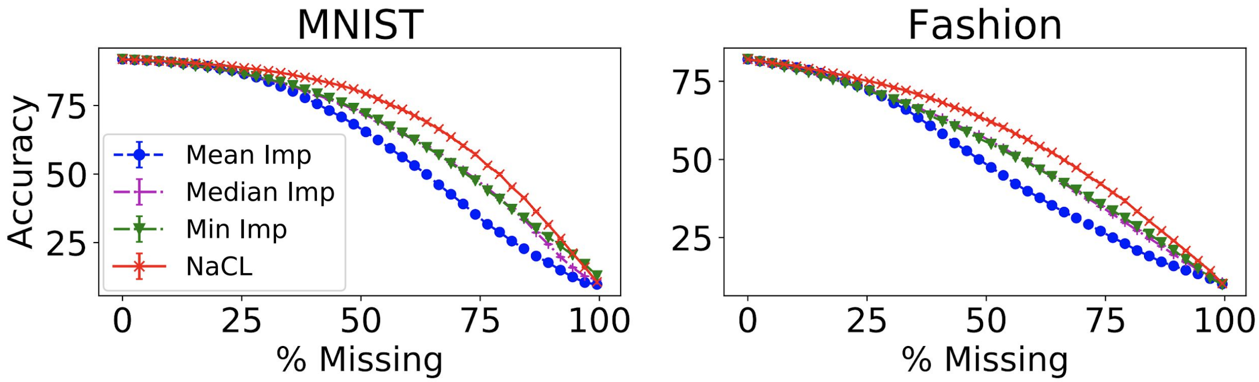
Consider **all possible complete inputs** and **reason** about the *expected* behavior of the classifier

$$\mathbb{E}_{\mathbf{x}^m \sim p(\mathbf{x}^m|\mathbf{x}^o)}\left[f\left(\mathbf{x}^m\mathbf{x}^o\right)\right]$$

x^{o} = observed features
x^{m} = missing features

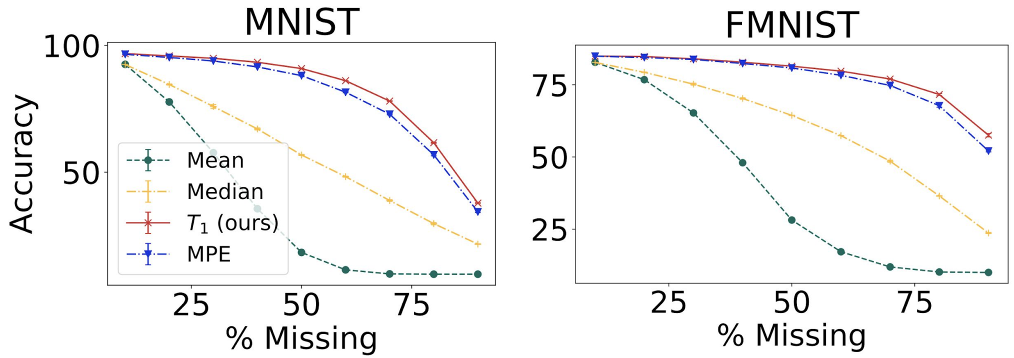
Experiment:

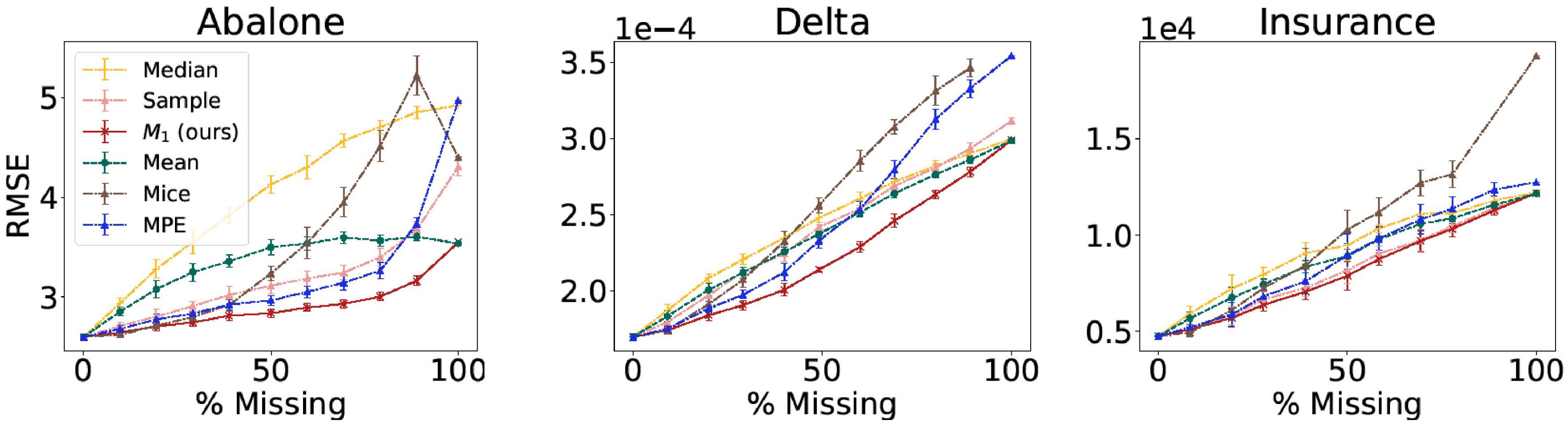
- *f(x)* = logistic regres.
- *p(x)* = naive Bayes



[Khosravi et al. IJCAI19, NeurIPS20, Artemiss20]

Probabilistic Circuits for Missing Data





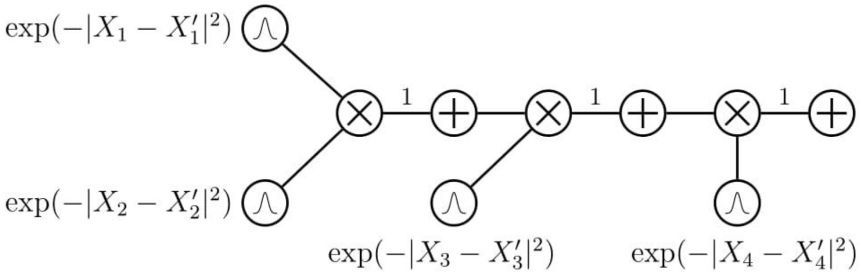
[Khosravi et al. IJCAI19, NeurIPS20, Artemiss20]

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

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

Model-Based Algorithmic Fairness: FairPC

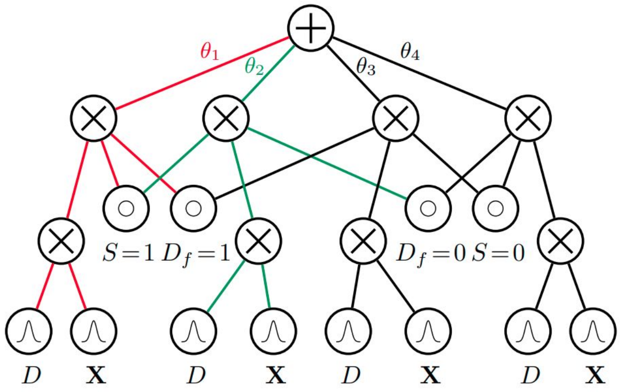
Learn classifier given

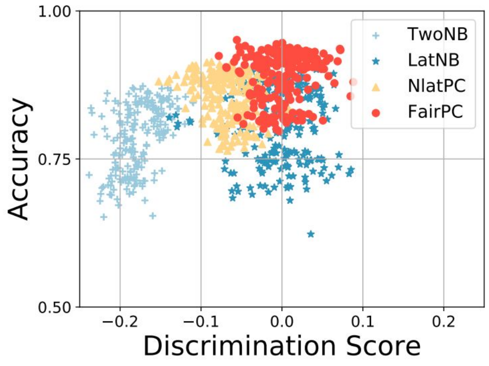
- features S and X
- training labels/decisions D

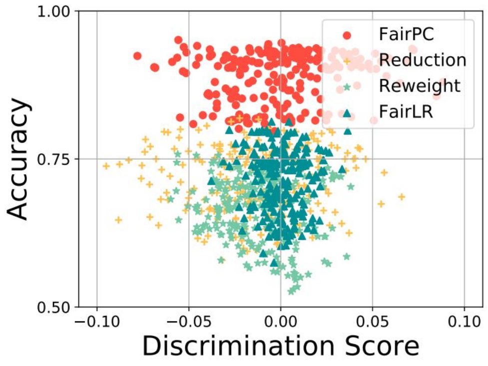
Group fairness by demographic parity:

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]

Probabilistic Sufficient Explanations

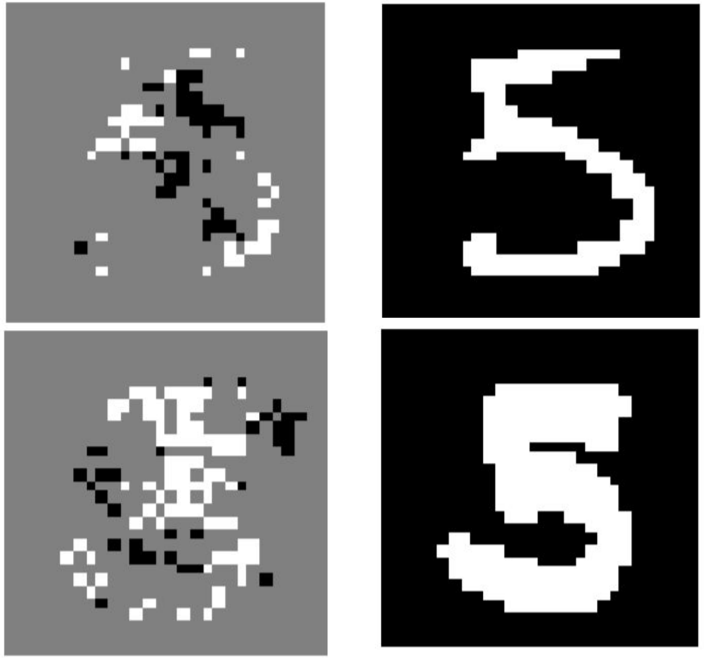
Goal: explain an instance of classification (a specific prediction)

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.

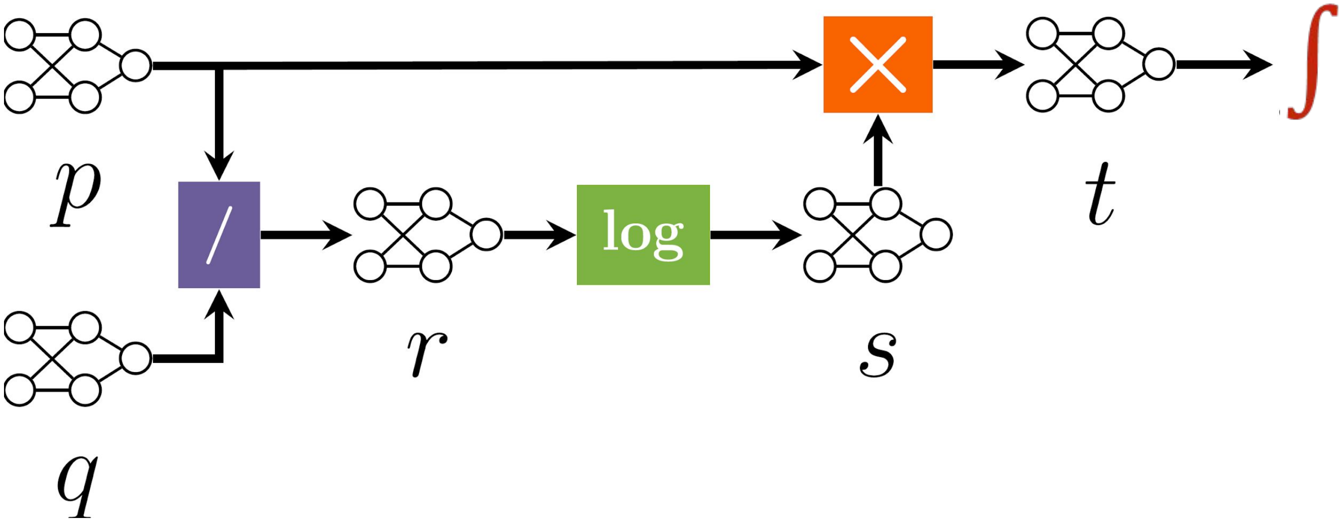
2. It is minimal and "simple"



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

Queries as pipelines: KLD

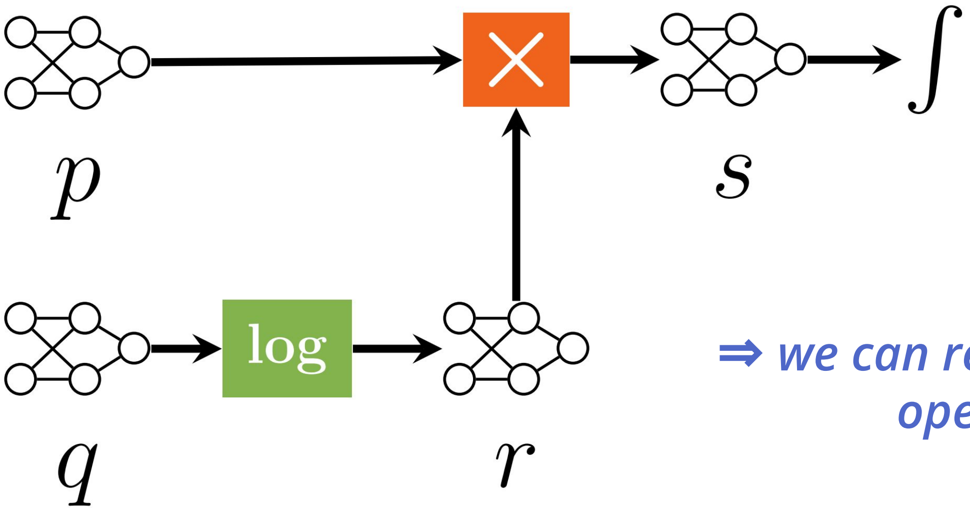
$$\mathbb{KLD}(p \parallel q) = \int p(\boldsymbol{x}) \times \log((p(\boldsymbol{x})/q(\boldsymbol{x}))d\boldsymbol{X}$$



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

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

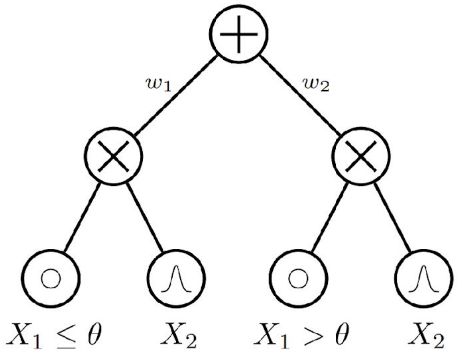


⇒ *we can reuse the operations!*

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.

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

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 $\boldsymbol{e}$,

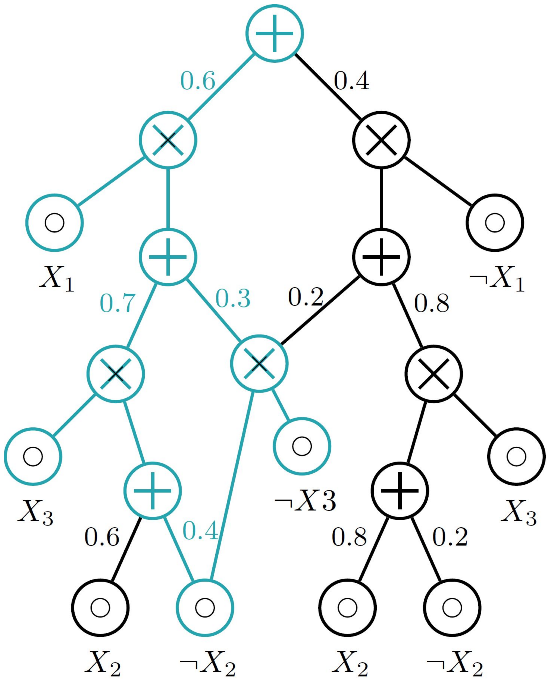
find: $argmax_{\boldsymbol{q}}\, p(\boldsymbol{q}|\boldsymbol{e})$

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

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

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

Pruning circuits



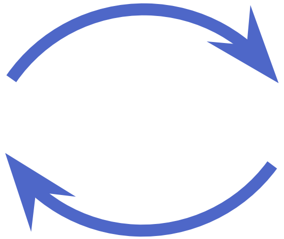
Any parts of circuit not relevant for MMAP state can be pruned away

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

We can find such edges in *linear time*

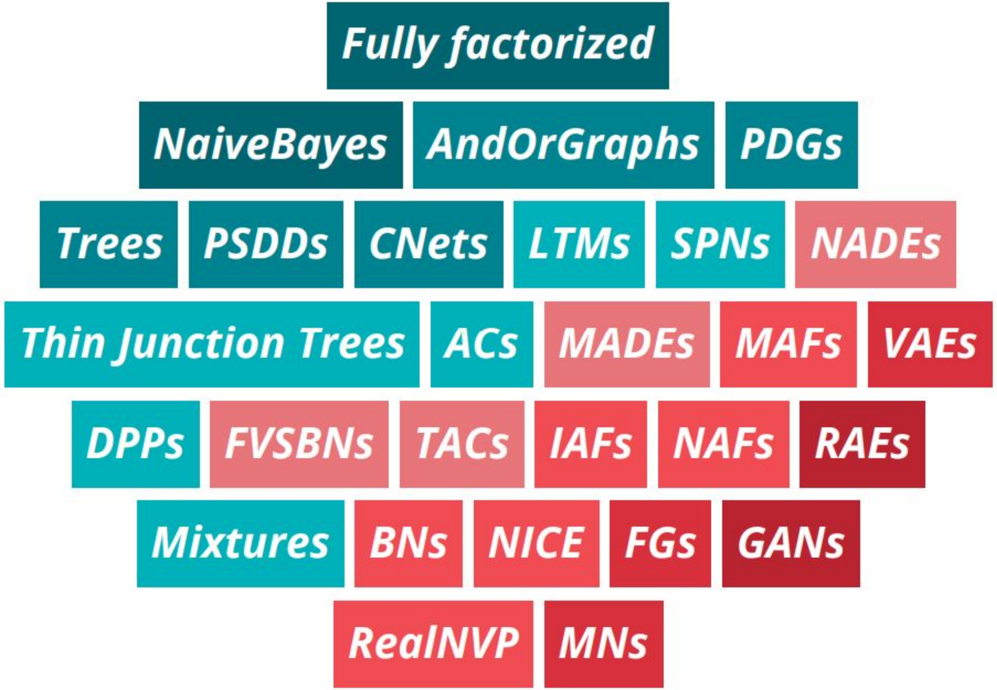
YooJung Choi, Tal Friedman and Guy Van den Broeck. Solving Marginal MAP Exactly by Probabilistic Circuit Transformations, 2021.

Iterative MMAP solver

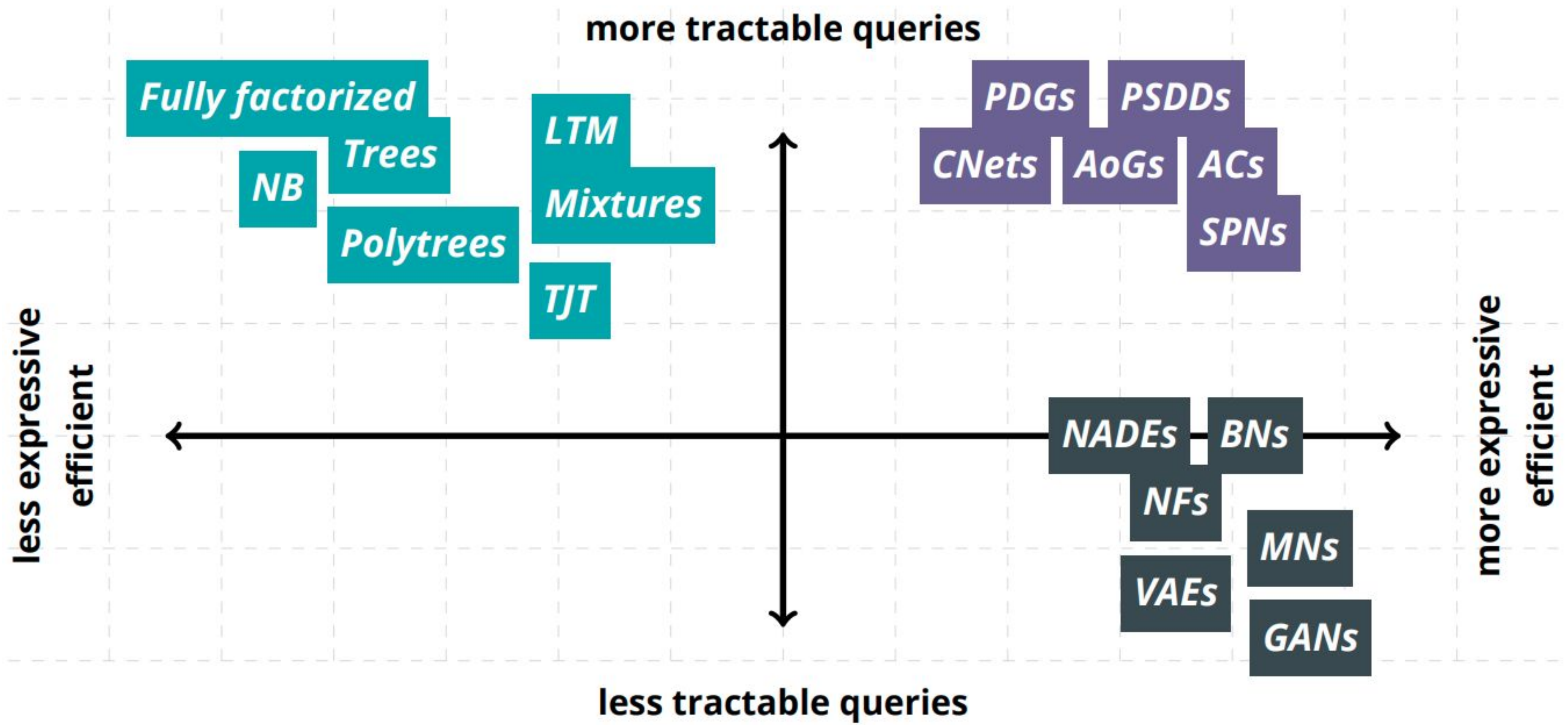


Dataset	runtime (# solved) search	pruning
NLTCS	**0.01** (10)	0.63 (10)
MSNBC	**0.03** (10)	0.73 (10)
KDD	**0.04** (10)	0.68 (10)
Plants	2.95 (10)	**2.72** (10)
Audio	2041.33 (6)	**13.70 (10)**
Jester	2913.04 (2)	**14.74 (10)**
Netflix	– (0)	**47.18 (10)**
Accidents	109.56 (10)	**15.86** (10)
Retail	**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)**

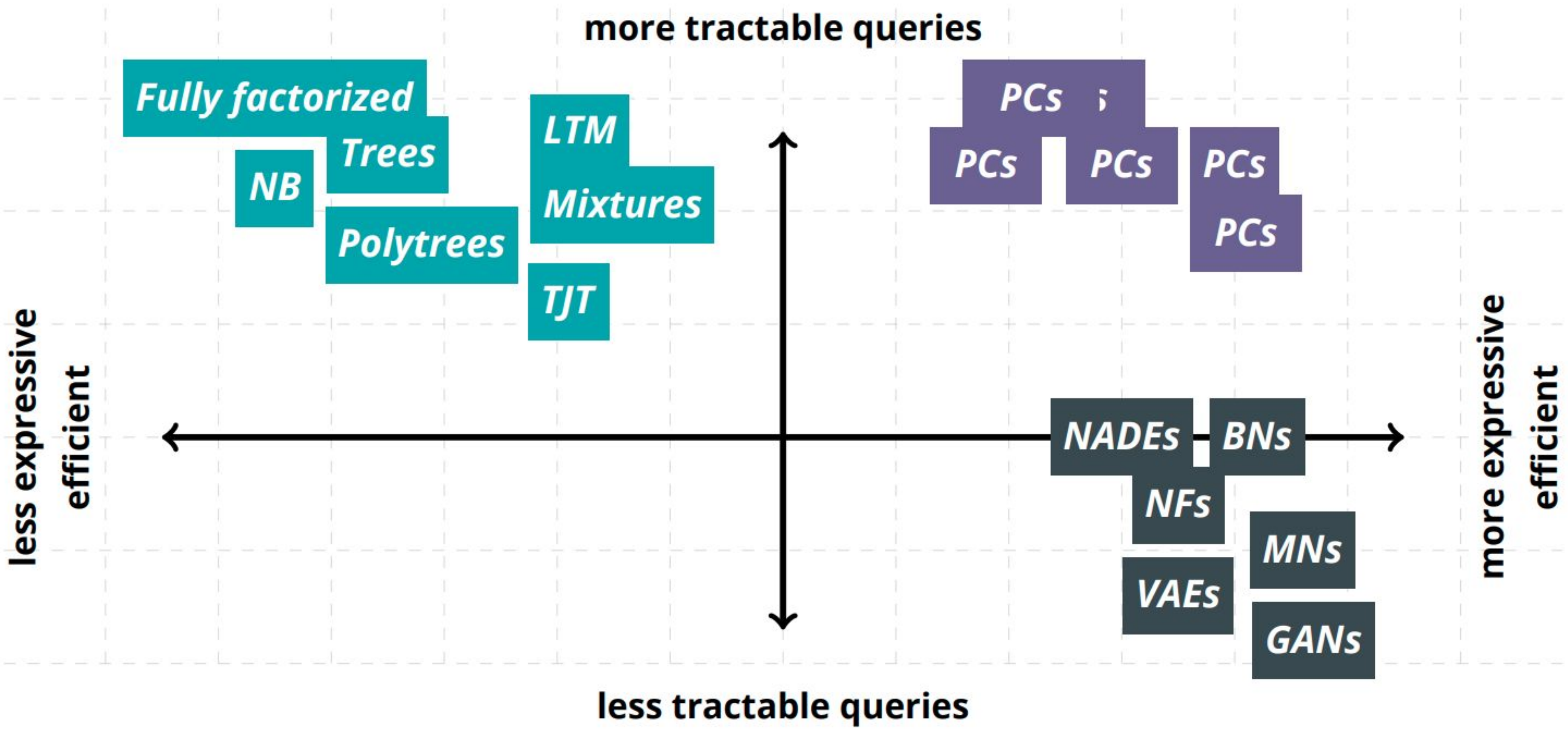
YooJung Choi, Tal Friedman and Guy Van den Broeck. Solving Marginal MAP Exactly by Probabilistic Circuit Transformations, 2021.



tractability is a spectrum



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

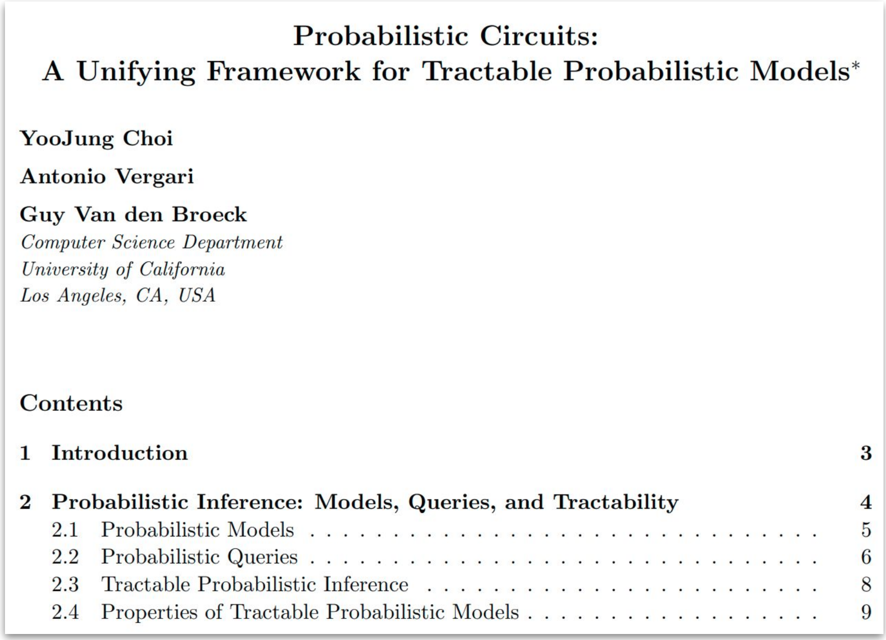

more tractable queries
Fully factorized
Trees
NB
Polytrees
LTM
Mixtures
TJT
PCs
PCs
PCs
PCs
PCs
less expressive
efficient
more expressive
efficient
NADEs
BNs
NFs
MNs
VAEs
GANs
less tractable queries

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

1 Introduction **3**

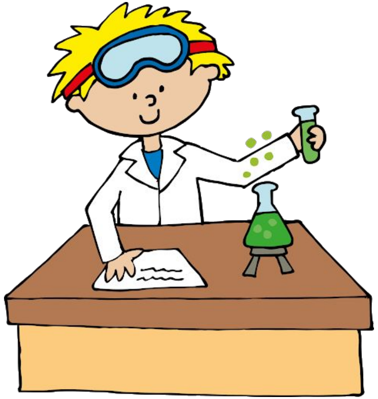
2 Probabilistic Inference: Models, Queries, and Tractability **4**
2.1 Probabilistic Models 5
2.2 Probabilistic Queries 6
2.3 Tractable Probabilistic Inference 8
2.4 Properties of Tractable Probabilistic Models 9

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

Probabilistic Programs

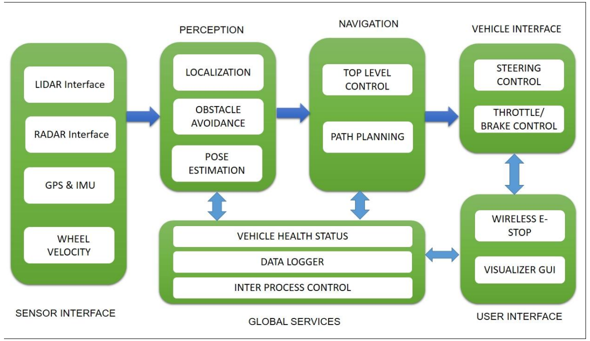
Motivation from the AI side:
Making modern AI systems is **too hard**

System Builders



Model Builders

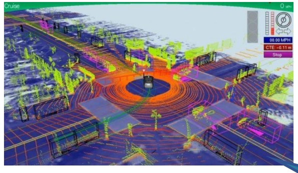
AI System Builder



Need to integrate uncertainty over the whole system

20% chance of obstacle!

94% chance of obstacle!



. . .

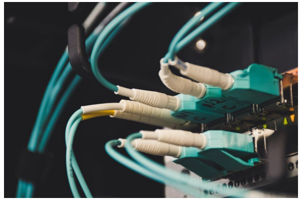
99% certain about current location

Inside the Self-Driving Tesla Fatal Accident

By ANJALI SINGHVI and KARL RUSSELL UPDATED July 12, 2016

The accident may have happened in part because the crash-avoidance system is designed to engage only when radar and computer vision systems agree that there is an obstacle, according to an industry executive with direct

AI Model Builder



“When you have the flu you have a cough 70% of the time”

“What is the probability that a patient with a fever has the flu?”

“Routers fail on average every 5 years”

“What is the probability that my packet will reach the target server?” [SGTVV SIGCOMM’20]

Probabilistic Programs

```
let x = flip 0.5 in
let y = flip 0.7 in
let z = x || y in
let w = if z then
      my_func(x,y)
else
      …
in
observe(z)
```

means "flip a coin, and output true with probability ½"

Standard (functional) programming constructs: let, if, ...

means "reject this execution if z is not true"

Why Probabilistic Programming?

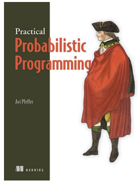
- PPLs are proliferating

Pyro

Edward

HackPPL

Stan



Figaro

Venture, Church, IBAL, WebPPL, Infer.NET, Tensorflow Probability, ProbLog, PRISM, LPADs, CPLogic, CLP(BN), ICL, PHA, Primula, Storm, Gen, PRISM, PSI, Bean Machine, etc. … *and many many more*

- Programming languages are humanity's biggest knowledge representation achievement!
- Programs should be AI models

Focus on Discrete Models

1. Real programs have inherently discrete structure (e.g. if-statements)
2. Discrete structure is inherent in many domains (graphs, text, ranking, etc.)
3. Many existing PPLs assume smooth and differentiable densities and do not handle discreteness well.

Stan

Does not support if-statements!

WebPPL

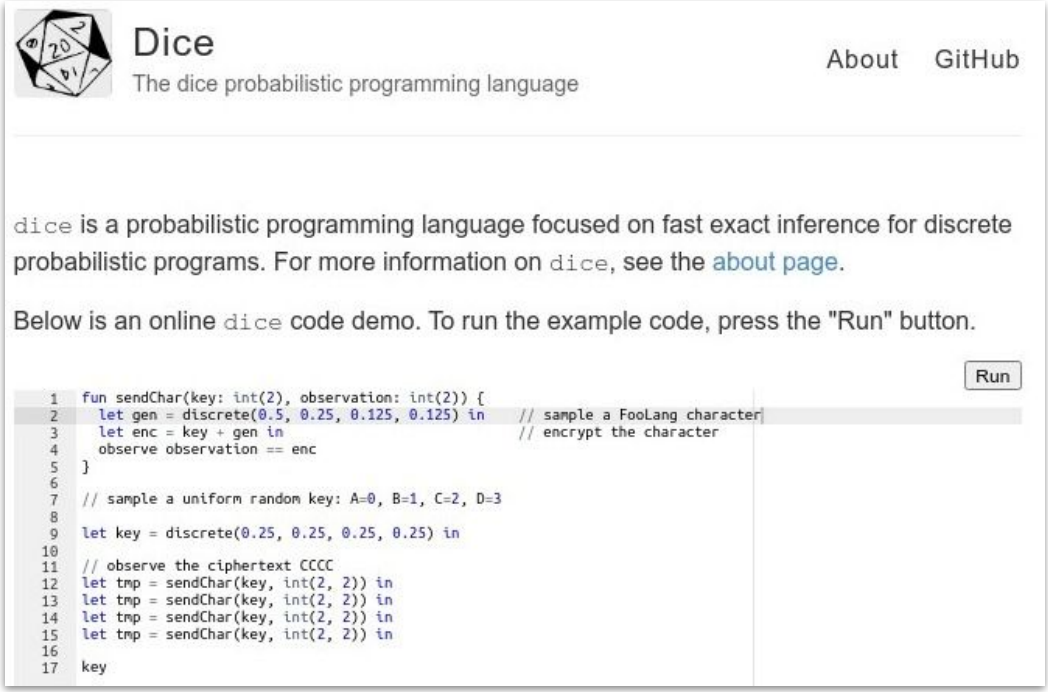
coroutines. Whenever a discrete variable is encountered in a program's execution, the program is suspended and resumed multiple times with all possible values in the support of that distribution. Listing 10, which impl[...]nts a simple finite

[AADB+'19]

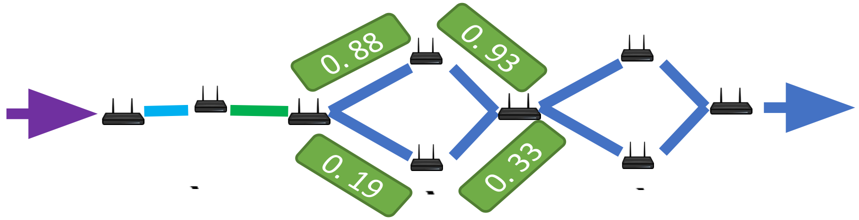
Discrete probabilistic programming is the important unsolved open problem!

Dice language for discrete probabilistic programs

http://dicelang.cs.ucla.edu/ [Holtzen et al. OOPSLA20]



Network Verification in Dice




```
fun n1(init: bool) {
   let l1succeed = flip 0.99 in
   let l2succeed = flip 0.91 in
   init && l1succeed && l2succeed
}
```

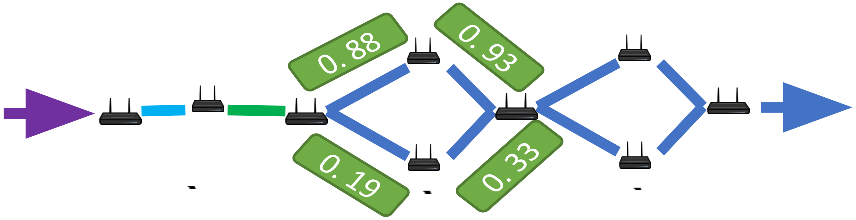
```
fun n2(init: bool) {
   let routeChoice = flip 0.5 in
   if routeChoice then
      init && flip 0.88 && flip 0.93
   else
      init && flip 0.19 && flip 0.33
}
```

ECMP equal-cost path protocol: choose randomly which router to forward to

Main routine, combines the networks

```
n2(n2(n1(true)))
```

Network Verification i



fun n1(
  let l1
  let l2
  init &
}

This doesn't show all the language features of dice:

- Integers
- Tuples
- Bounded recursion
- Bayesian conditioning
- …

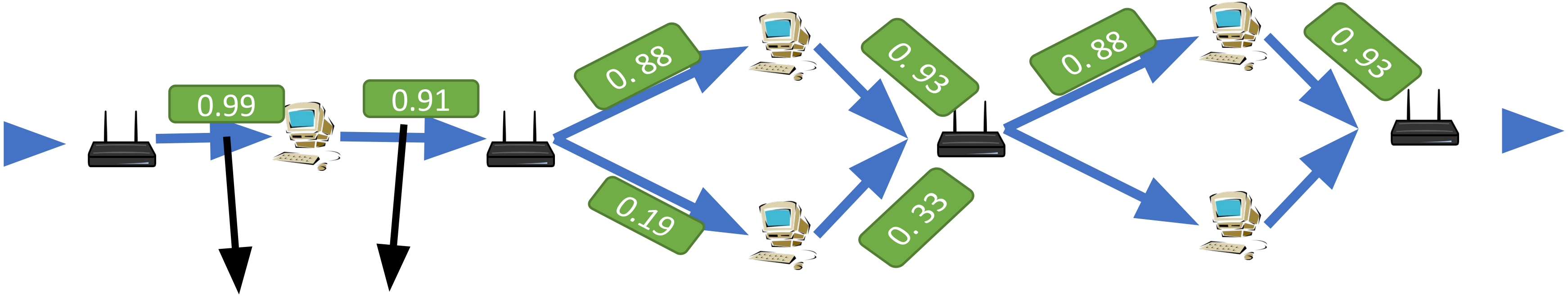
```
fun n2(init: bool) {
  let routeChoice = flip 0.5 in
  if routeChoice then
    init && flip 0.88 && flip 0.93
  else
    init && flip 0.19 && flip 0.33
}
```

ECMP equal-cost path protocol: choose randomly which router to forward to

Main routine, combines the networks

n2(n2(n1(true)))

Probabilistic Program Inference



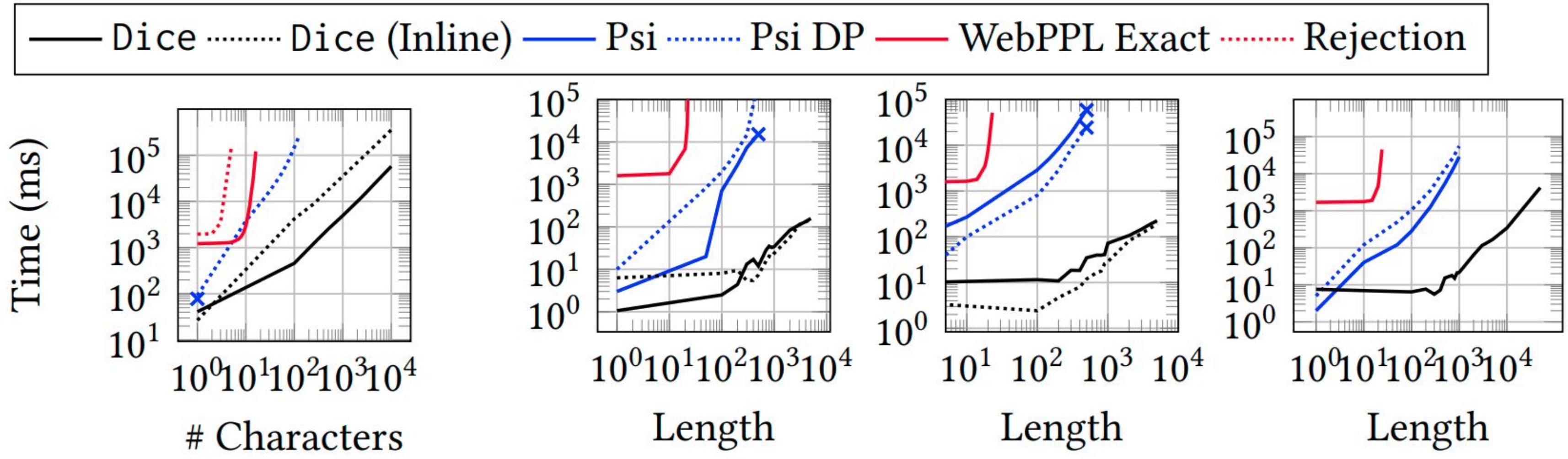
0.99 x 0.91 x 0.5 x 0.88 x 0.93 x 0.5 x 0.88 x 0.93

+ 0.99 x 0.91 x 0.5 x 0.19 x 0.33 x 0.5 x 0.88 x 0.93

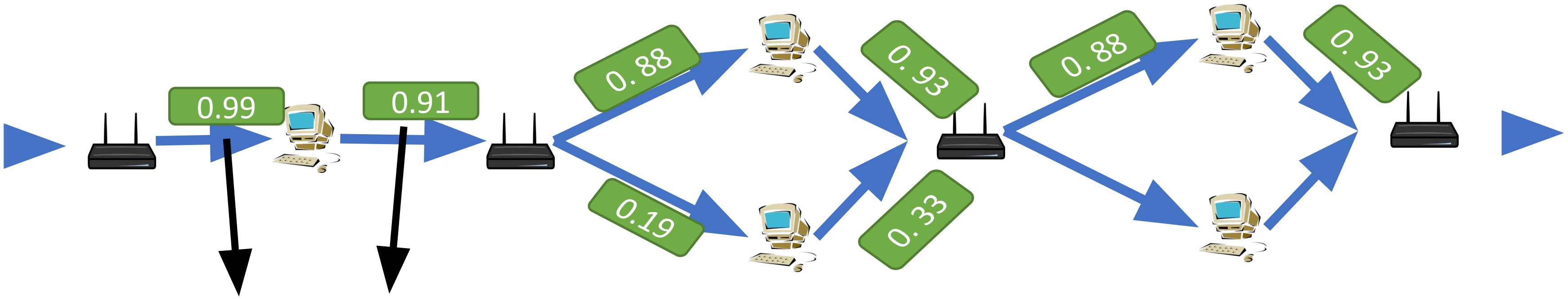
+ ...

Probabilistic Program Inference

Path enumeration: find all of them!



Key to Fast Inference: **Factorization** (product nodes)



$$\boxed{\begin{aligned} &0.99 \times 0.91 \\ + \; &0.99 \times 0.91 \\ + \; &\ldots \end{aligned}} \begin{aligned} &\times 0.5 \times 0.88 \times 0.93 \times 0.5 \times 0.88 \times 0.93 \\ &\times 0.5 \times 0.19 \times 0.33 \times 0.5 \times 0.88 \times 0.93 \\ & \end{aligned}$$

Easy to see on the graph structure …
how about on the program?

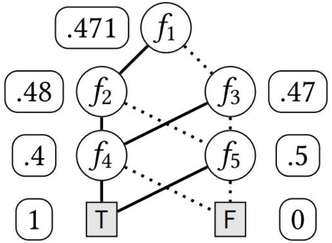
Symbolic Compilation in Dice

- Construct Boolean formula
- Satisfying assignments ≈ paths
- Variables are flips
- Associate weights with flips
- Compile factorized circuit

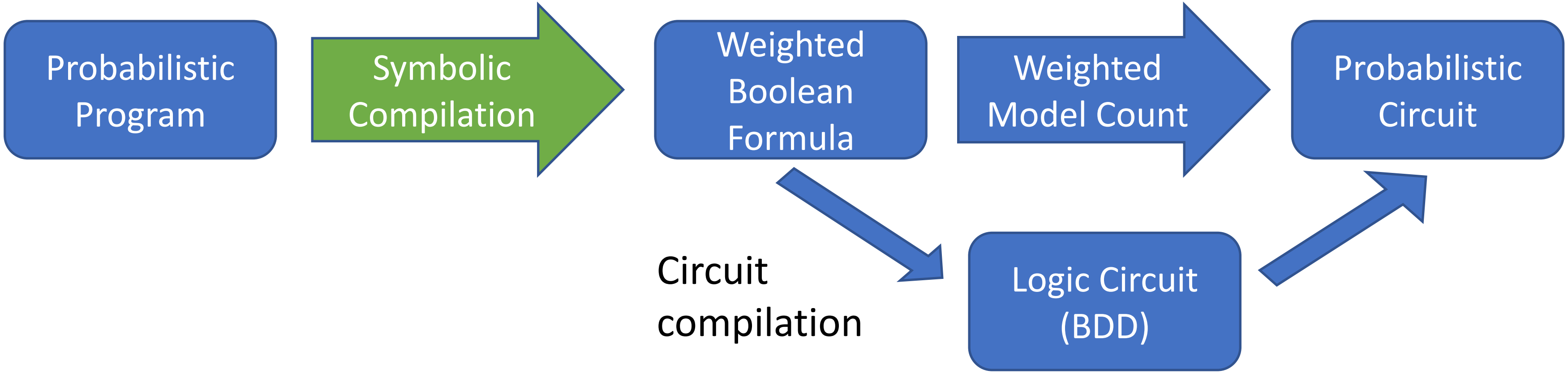
```
let x = flip₁ 0.1 in
let y = if x then flip₂ 0.2 else
    flip₃ 0.3 in
let z = if y then flip₄ 0.4 else
    flip₅ 0.5 in z
```

$$\underbrace{0.1}_{x=T} \cdot \underbrace{0.2}_{y=T} \cdot \underbrace{0.4}_{z=T} + \underbrace{0.1}_{x=T} \cdot \underbrace{0.8}_{y=F} \cdot \underbrace{0.5}_{z=T} + \underbrace{0.9}_{x=F} \cdot \underbrace{0.3}_{y=T} \cdot \underbrace{0.4}_{z=T} + \underbrace{0.9}_{x=F} \cdot \underbrace{0.7}_{y=F} \cdot \underbrace{0.5}_{z=T}$$

$$\Rightarrow \quad f_1 f_2 f_4 \vee f_1 \bar{f}_2 f_5 \vee \bar{f}_1 f_3 f_4 \vee \bar{f}_1 \bar{f}_3 f_5 \quad \Rightarrow$$

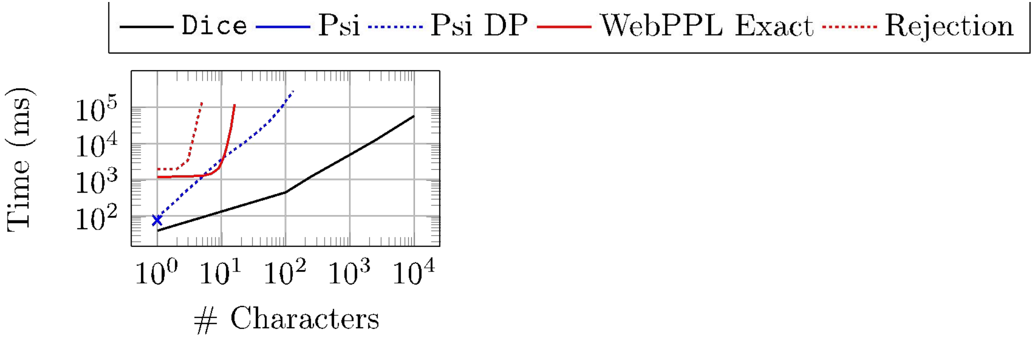


Symbolic Compilation in Dice to **Probabilistic Circuits**



Experimental Evaluation

- Example from text analysis: breaking a Caesar cipher



- Competitive with specialized Bayesian network solvers

More program paths than atoms in the universe

Benchmark	Psi (ms)	DP (ms)	Dice (ms)	# Parameters	# Paths	BDD Size
Cancer	772	46	**13**	10	1.1×10^3	28
Survey	2477	152	**13**	21	1.3×10^4	73
Alarm	✗	✗	**25**	509	1.0×10^{36}	1.3×10^3
Insurance	✗	✗	**212**	984	1.2×10^{40}	1.0×10^5
Hepar2	✗	✗	**54**	48	2.9×10^{69}	1.3×10^3
Hailfinder	✗	✗	**618**	2656	2.0×10^{76}	6.5×10^4
Pigs	✗	✗	**72**	5618	7.3×10^{492}	35
Water	✗	✗	**2590**	1.0×10^4	3.2×10^{54}	5.1×10^4
Munin	✗	✗	**1866**	8.1×10^5	2.1×10^{1622}	1.1×10^4

Better Inference. How?

Exploit modularity - program structure

1. AI modularity:
   Discover contextual independencies and **factorize**
2. PL modularity:
   Compile procedure summaries and reuse at each call site

Reason about programs! Compiler optimizations:

3. Flip hoisting optimization
4. Determinism, optimize integer representation, etc.

Flip Hoisting

```
let x = flip 0.1 in
let z = flip 0.2 in
let y = if x && z then flip 0.3
    else if x && !z then flip 0.4
    else flip 0.3
in y
```

≡

```
let x = flip 0.1 in let z = flip 0.2 in
let tmp = flip 0.3 in
let y = if x && z then tmp
    else if x && !z then flip 0.4
    else tmp
in y
```

- Fewer flips = smaller compiled circuits = faster
- But, be careful with soundness:

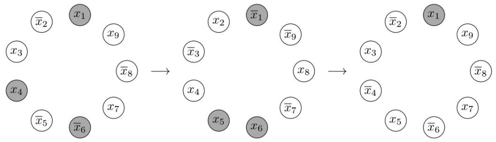
```
flip 0.3 && flip 0.3
```

≢

```
let tmp = flip 0.3 in tmp && tmp;
```

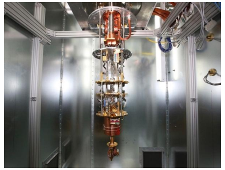
If you build it they will come

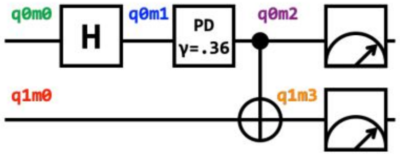
As soon as ***dice*** was put online people started using it in surprising ways we had not foreseen



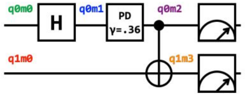
Probabilistic Model Checking

Prism → Rubicon → dice





Quantum Simulation



quantum circuit

probabilistic circuit

If you build it they will come

In both cases, ***dice*** outperforms existing specialized methods on important examples!

Probabilistic Model Checking

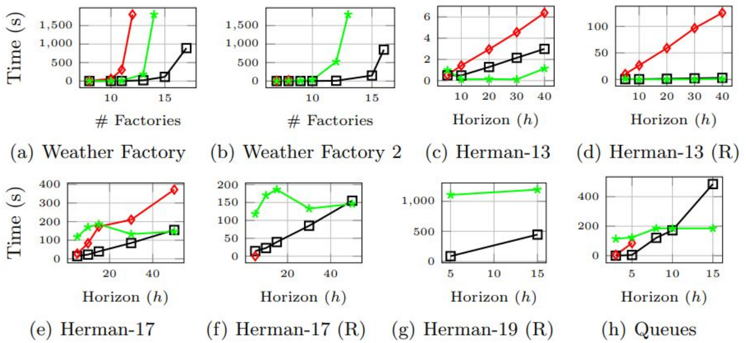
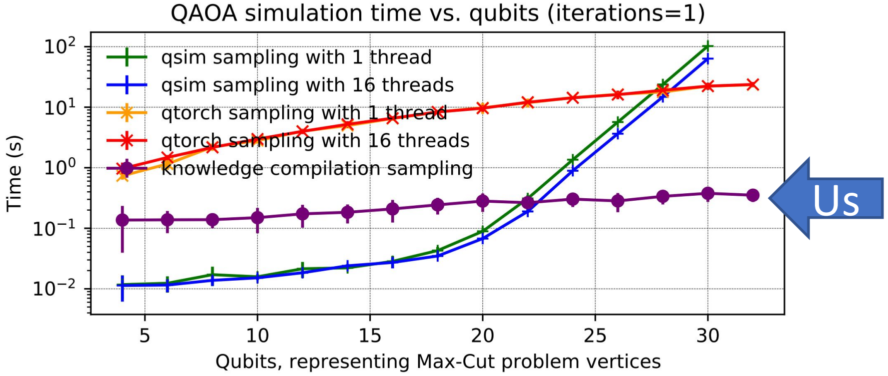


Fig. 9. Scaling plots comparing RUBICON (—□—), STORM's symbolic engine (—◇—), and STORM's explicit engine (—★—). An "(R)" in the caption denotes random parameters.

Quantum Simulation



Competitive with well-known simulators like Google qsim and qtorch [FSC+ PloS one '18] !

Computational Abstractions

Let us think of probability as something that is computed.

Abstraction = Structure of Computation

Two levels of abstraction:

Probabilistic Programs — “High-level code”

Probabilistic Circuits — “Machine code”

Thanks

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

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