Logic & Probabilistic Circuits

Representation
Reasoning
Theory

Guy Van den Broeck
University of California, Los Angeles
guyvdb@cs.ucla.edu

YooJung Choi
Arizona State University
yj.choi@asu.edu

*August 22nd, 2023 - **Logic and Algorithms in Database Theory and AI Boot Camp** @ Simons Institute*

Objective

- Circuits are an assembly language for tractable logic and probabilistic reasoning
- Even though logic is central to this Simons program,
  we will couch this tutorial in probability...
  - Most AI and DB interest in tractable logic circuits for the past 15 years has been as a means of doing probabilistic inference
  - Much richer query languages <3
  - We live in the age of probabilistic *generative AI*... :-)
- We will spare you most of the machine learning details, and instead focus on representations, query languages, reasoning algorithms, and connections to theory.

Objective

- Circuits are an assembly language for tractable logic and probabilistic reasoning
- Even though logic is central to this Simons program,
  we will couch this tutorial in probability...
  - Most AI and DB interest in tractable logic circuits for the past 15 years has been as a means of doing probabilistic inference
  - Much richer query languages <3
  - We live in the age of probabilistic *generative AI*... :-)
- We will spare you most of the machine learning details, and instead focus on representations, query languages, reasoning algorithms, and connections to theory.

Objective

- Circuits are an assembly language for tractable logic and probabilistic reasoning
- Even though logic is central to this Simons program,
  we will couch this tutorial in probability...
  - Most AI and DB interest in tractable logic circuits for the past 15 years has been as a means of doing probabilistic inference
  - Much richer query languages <3
  - We live in the age of probabilistic *generative AI*... :-)
- We will spare you most of the machine learning details, and instead focus on representations, query languages, reasoning algorithms, and connections to theory.

Questions to be answered today

1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. *Are all tractable distributions probabilistic circuits?* (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

Questions to be answered today

1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. *Are all tractable distributions probabilistic circuits?* (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

Questions to be answered today

1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. *Are all tractable distributions probabilistic circuits?* (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

Questions to be answered today

1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. *Are all tractable distributions probabilistic circuits?* (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

Questions to be answered today

1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. *Are all tractable distributions probabilistic circuits?* (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

Questions to be answered today

1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. *Are all tractable distributions probabilistic circuits?* (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

Questions to be answered today

1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. *Are all tractable distributions probabilistic circuits?* (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

Questions to be answered today

1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. *Are all tractable distributions probabilistic circuits?* (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

Acknowledgements

This tutorial is based on our (joint) tutorials and slides from

- Antonio Vergari
- Robert Peharz
- Nicola Di Mauro
- Honghua Zhang
- Benjie Wang

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 GPT

The *Alphabet Soup* of probabilistic 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* *GPT*

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* *GPT*

tractability is a spectrum

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* *GPT*

Expressive **models without** ***compromises***

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* *GPT*

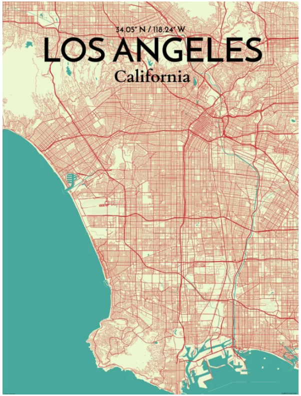
a ***unifying framework*** for tractable models

Why tractable inference?

or the inherent trade-off of tractability vs. expressiveness

Why probabilistic inference?

q_1: *What is the probability that today is a Monday and there is a traffic jam on Westwood Blvd.?*

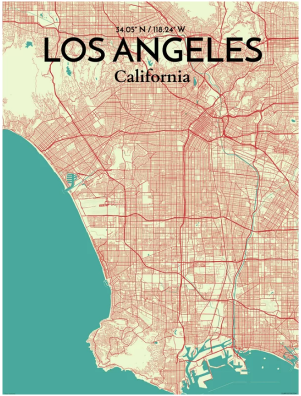


Why probabilistic inference?

$\mathbf{q}_1$: *What is the probability that today is a Monday and there is a traffic jam on Westwood Blvd.?*

$$\mathbf{X} = \{\mathsf{Day}, \mathsf{Time}, \mathsf{Jam}_{\mathsf{Str1}}, \mathsf{Jam}_{\mathsf{Str2}}, \ldots, \mathsf{Jam}_{\mathsf{StrN}}\}$$

$$\mathbf{q}_1(\mathbf{m}) = p_{\mathbf{m}}(\mathsf{Day} = \mathsf{Mon}, \mathsf{Jam}_{\mathsf{Wwood}} = 1)$$



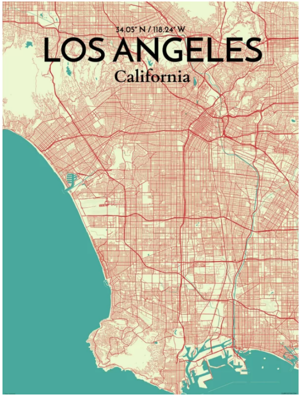
Why probabilistic inference?

q_1: *What is the probability that today is a Monday and there is a traffic jam on Westwood Blvd.?*

$$\mathbf{X} = \{\text{Day}, \text{Time}, \text{Jam}_{\text{Str1}}, \text{Jam}_{\text{Str2}}, \ldots, \text{Jam}_{\text{StrN}}\}$$

$$\mathrm{q}_1(\mathbf{m}) = p_{\mathbf{m}}(\text{Day} = \text{Mon}, \text{Jam}_{\text{Wwood}} = 1)$$

$\Longrightarrow$ ***marginals***



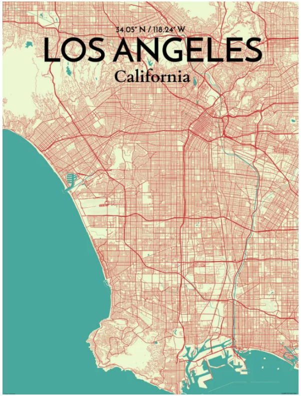
© fineartamerica.com

Why probabilistic inference?

$\mathbf{q}_2$: *Which day is most likely to have a traffic jam on my route to campus?*

$$\mathbf{X} = \{\mathsf{Day}, \mathsf{Time}, \mathsf{Jam}_{\mathsf{Str1}}, \mathsf{Jam}_{\mathsf{Str2}}, \ldots, \mathsf{Jam}_{\mathsf{StrN}}\}$$

$$\mathbf{q}_2(\mathbf{m}) = \operatorname{argmax}_{\mathsf{d}}\, p_{\mathbf{m}}(\mathsf{Day} = \mathsf{d} \wedge \bigvee_{i \in \mathsf{route}} \mathsf{Jam}_{\mathsf{Str}i})$$



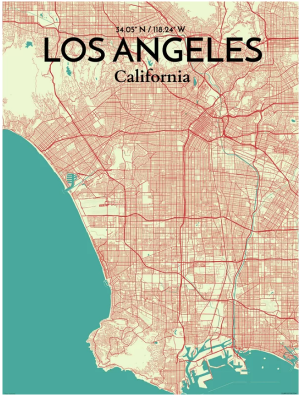
Why probabilistic inference?

q_2: *Which day is most likely to have a traffic jam on my route to campus?*

$$\mathbf{X} = \{\mathsf{Day}, \mathsf{Time}, \mathsf{Jam}_{\mathsf{Str1}}, \mathsf{Jam}_{\mathsf{Str2}}, \ldots, \mathsf{Jam}_{\mathsf{StrN}}\}$$

$$\mathrm{q}_2(\mathbf{m}) = \operatorname{argmax}_{\mathsf{d}}\, p_{\mathbf{m}}(\mathsf{Day} = \mathsf{d} \wedge \bigvee_{i \in \mathsf{route}} \mathsf{Jam}_{\mathsf{Str}i})$$

$\Longrightarrow$ ***marginals + MAP + logical events***



© fineartamerica.com

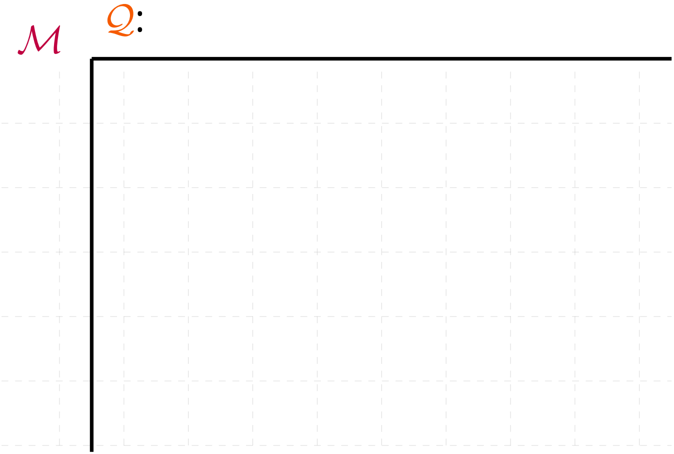
Tractable Probabilistic Inference

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* $O(\mathrm{poly}(|\mathbf{m}|))$.

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* $O(\mathsf{poly}(|\mathbf{m}|))$.

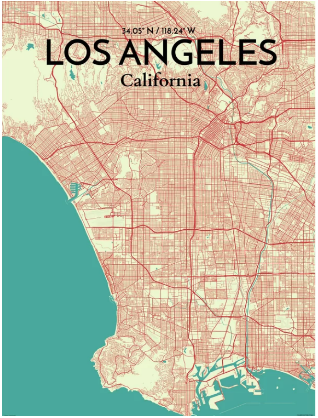
$\Longrightarrow$ *often* poly *will in fact be* ***linear****!*



tractable bands

Complete evidence (EVI)

q_3: *What is the probability that today is a Monday at 12.00 and there is a traffic jam only on Westwood Blvd.?*

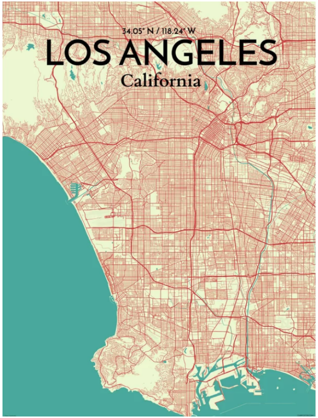


Complete evidence (EVI)

$\mathbf{q}_3$: *What is the probability that today is a Monday at 12.00 and there is a traffic jam only on Westwood Blvd.?*

$$\mathbf{X} = \{\mathsf{Day}, \mathsf{Time}, \mathsf{Jam}_{\mathsf{Wwood}}, \mathsf{Jam}_{\mathsf{Str2}}, \ldots, \mathsf{Jam}_{\mathsf{StrN}}\}$$

$$\mathbf{q}_3(\mathbf{m}) = p_{\mathbf{m}}(\mathbf{X} = \{\mathsf{Mon}, 12.00, 1, 0, \ldots, 0\})$$



Complete evidence (EVI)

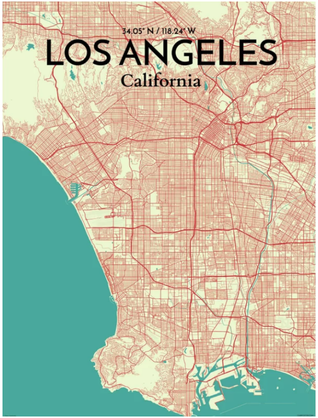
q_3: *What is the probability that today is a Monday at 12.00 and there is a traffic jam only on Westwood Blvd.?*

$$\mathbf{X} = \{\text{Day}, \text{Time}, \text{Jam}_{\text{Wwood}}, \text{Jam}_{\text{Str2}}, \ldots, \text{Jam}_{\text{StrN}}\}$$

$$q_3(\mathbf{m}) = p_{\mathbf{m}}(\mathbf{X} = \{\text{Mon}, 12.00, 1, 0, \ldots, 0\})$$

...fundamental in ***maximum likelihood learning***

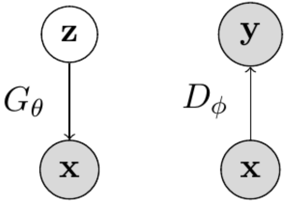
$$\theta_{\mathbf{m}}^{\text{MLE}} = \operatorname{argmax}_{\theta} \prod_{\mathbf{x} \in \mathcal{D}} p_{\mathbf{m}}(\mathbf{x}; \theta)$$



© fineartamerica.com

Generative Adversarial Networks

$$\min_\theta \max_\phi \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[\log D_\phi(\mathbf{x})\right] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log(1 - D_\phi(G_\theta(\mathbf{z})))\right]$$

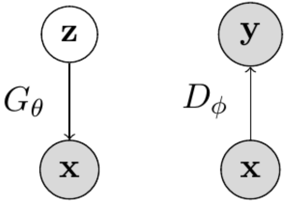


Goodfellow et al., "Generative adversarial nets", 2014

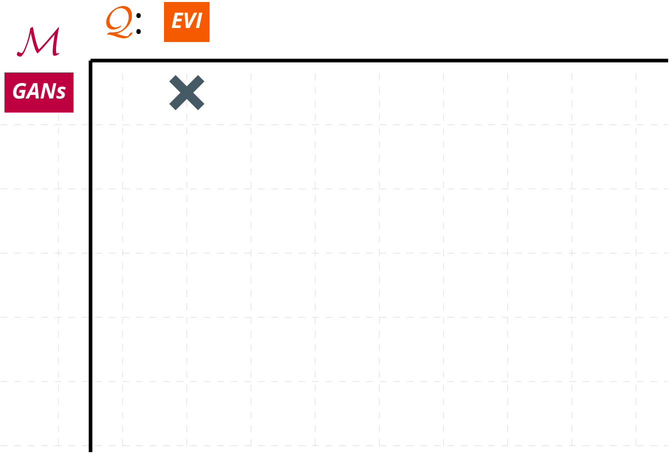
~~Generative Adversarial Networks~~

$$\min_\theta \max_\phi \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[\log D_\phi(\mathbf{x})\right] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log(1 - D_\phi(G_\theta(\mathbf{z})))\right]$$

- no explicit likelihood!
  $\Longrightarrow$ *adversarial training instead of MLE*
  $\Longrightarrow$ *no tractable EVI*
- good sample quality
  $\Longrightarrow$ *but lots of samples needed for MC*
- unstable training $\Longrightarrow$ *mode collapse*



Goodfellow et al., "Generative adversarial nets", 2014

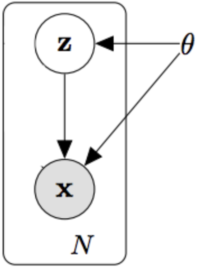


tractable bands

Variational Autoencoders

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z}$$

- an explicit likelihood model!



Rezende et al., "Stochastic backprop. and approximate inference in deep generative models", 2014
Kingma and Welling, "Auto-Encoding Variational Bayes", 2014

~~Variational Autoencoders~~

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x} \mid \mathbf{z})\right] - \mathbb{KL}(q_\phi(\mathbf{z} \mid \mathbf{x})||p(\mathbf{z}))$$

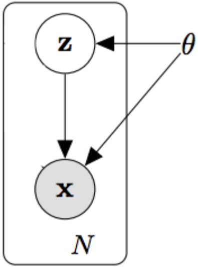
- an explicit likelihood model!
- ... but computing $\log p_\theta(\mathbf{x})$ is intractable

$\implies$ *an infinite and uncountable mixture*

$\implies$ *no tractable EVI*

- we need to optimize the ELBO...

$\implies$ *which is "tricky"*



$\mathcal{M}$	$\mathcal{Q}$: EVI
GANs	✗
VAEs	✗

tractable bands

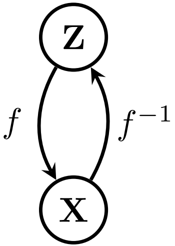
Normalizing flows

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Z}}(f^{-1}(\mathbf{x}))\left|\det\left(\frac{\delta f^{-1}}{\delta \mathbf{x}}\right)\right|$$

- an explicit likelihood!

$\Rightarrow$ *tractable EVI queries!*

- many neural variants
  - RealNVP *(Dinh et al. 2016)*, MAF *(Papamakarios et al. 2017)*
  - MADE *(Germain et al. 2015)*, PixelRNN *(Oord et al. 2016)*



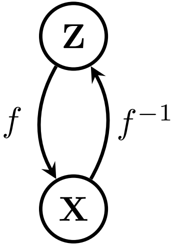
Normalizing flows

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Z}}(f^{-1}(\mathbf{x})) \left| \det \left(\frac{\delta f^{-1}}{\delta \mathbf{x}} \right) \right|$$

- an explicit likelihood!

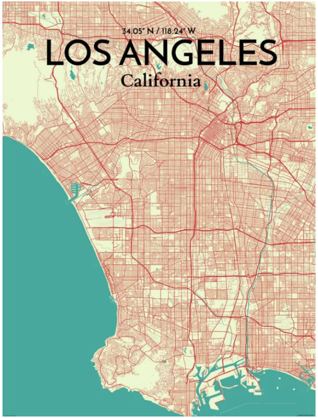
  $\Rightarrow$ *tractable EVI queries!*

- many neural variants
  - RealNVP *(Dinh et al. 2016)*, MAF *(Papamakarios et al. 2017)*
  - MADE *(Germain et al. 2015)*, PixelRNN *(Oord et al. 2016)*



Marginal queries (MAR)

q_1: *What is the probability that today is a Monday ~~at 12:00~~ and there is a traffic jam ~~only~~ on Westwood Blvd.?*

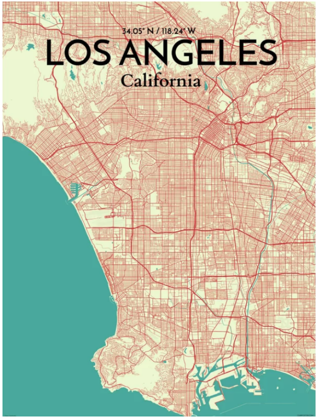


© fineartamerica.com

Marginal queries (MAR)

$\mathbf{q}_1$: *What is the probability that today is a Monday ~~at 12.00~~ and there is a traffic jam ~~only~~ on Westwood Blvd.?*

$$\mathbf{q}_1(\mathbf{m}) = p_{\mathbf{m}}(\mathsf{Day} = \mathsf{Mon}, \mathsf{Jam}_{\mathsf{Wwood}} = 1)$$



© fineartamerica.com

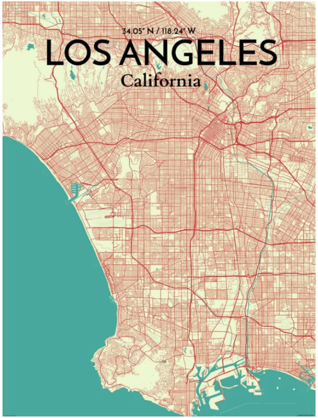
Marginal queries (MAR)

$\mathbf{q}_1$: *What is the probability that today is a Monday ~~at~~ ~~12.00~~ and there is a traffic jam ~~only~~ on Westwood Blvd.?*

$$\mathbf{q}_1(\mathbf{m}) = p_{\mathbf{m}}(\text{Day} = \text{Mon}, \text{Jam}_{\text{Wwood}} = 1)$$

General: $p_{\mathbf{m}}(\mathbf{e}) = \int p_{\mathbf{m}}(\mathbf{e}, \mathbf{H})\, d\mathbf{H}$

where $\mathbf{E} \subset \mathbf{X}, \quad \mathbf{H} = \mathbf{X} \setminus \mathbf{E}$



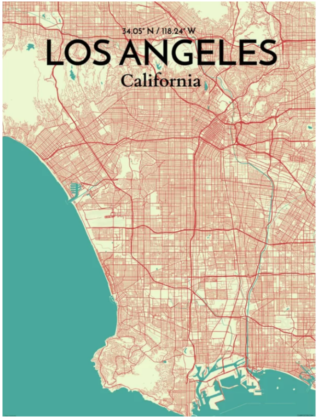
Marginal queries (MAR)

$\mathbf{q}_1$: *What is the probability that today is a Monday ~~at 12.00~~ and there is a traffic jam ~~only~~ on Westwood Blvd.?*

$$\mathbf{q}_1(\mathbf{m}) = p_{\mathbf{m}}(\mathsf{Day} = \mathsf{Mon}, \mathsf{Jam}_{\mathsf{Wwood}} = 1)$$

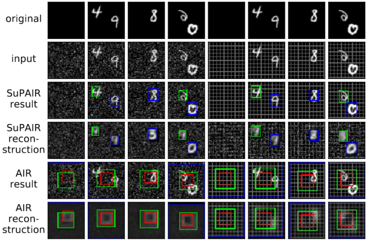
tractable MAR $\implies$ tractable ***conditional queries*** (CON):

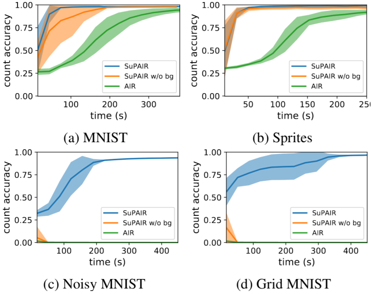
$$p_{\mathbf{m}}(\mathbf{q} \mid \mathbf{e}) = \frac{p_{\mathbf{m}}(\mathbf{q}, \mathbf{e})}{p_{\mathbf{m}}(\mathbf{e})}$$



© fineartamerica.com

Tractable MAR : scene understanding





Fast and exact marginalization over unseen or "do not care" parts in the scene

---

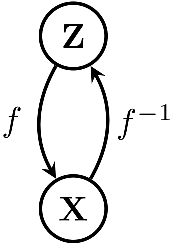
Stelzner et al., "Faster Attend-Infer-Repeat with Tractable Probabilistic Models", 2019
Kossen et al., "Structured Object-Aware Physics Prediction for Video Modeling and Planning", 2019

Normalizing flows

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Z}}(f^{-1}(\mathbf{x})) \left|\det\left(\frac{\delta f^{-1}}{\delta \mathbf{x}}\right)\right|$$

- an explicit likelihood!

$\Rightarrow$ *tractable EVI queries!*



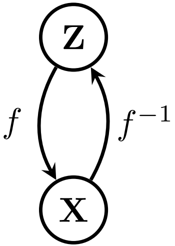
~~Normalizing flows~~

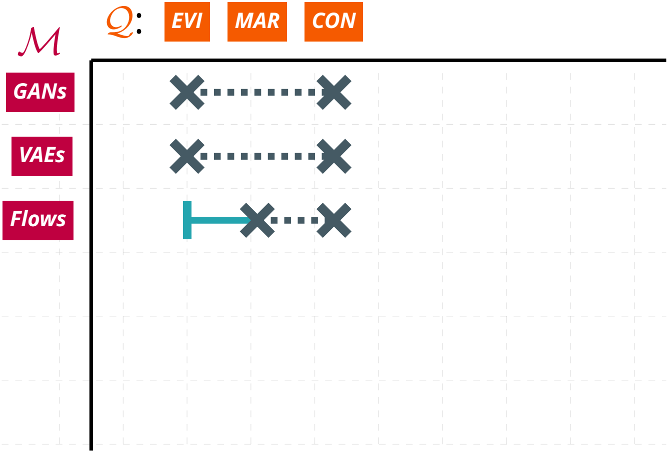
$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Z}}(f^{-1}(\mathbf{x}))\left|\det\left(\frac{\delta f^{-1}}{\delta \mathbf{x}}\right)\right|$$

- an explicit likelihood!

  $\Longrightarrow$ *tractable EVI queries!*

- ***MAR is generally intractable***: we cannot easily integrate over high-dimensional f



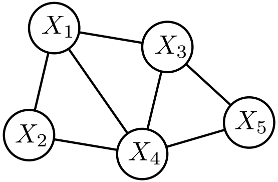


tractable bands

Probabilistic Graphical Models (PGMs)

Declarative semantics: a clean separation of modeling assumptions from inference

Nodes: random variables
Edges: dependencies



+

Inference:

- conditioning *(Darwiche 2001; Sang et al. 2005)*
- elimination *(Zhang et al. 1994; Dechter 1998)*
- message passing *(Yedidia et al. 2001; Dechter et al. 2002; Choi et al. 2010; Sontag et al. 2011)*

Complexity of MAR on PGMs

Exact complexity: Computing MAR and CON is *#P-hard*

$\Longrightarrow$ *(Cooper 1990; Roth 1996)*

Approximation complexity: Computing MAR and CON approximately within a relative error of $2^{n^{1-\epsilon}}$ for any fixed ϵ is *NP-hard*

$\Longrightarrow$ *(Dagum et al. 1993; Roth 1996)*

Treewidth!

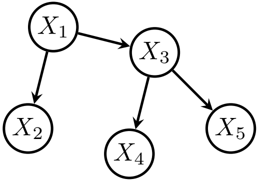
Treewidth:

Informally, how tree-like is the graphical model $\mathbf{m}$?

Fixed-parameter tractable: MAR and CON on a graphical model $\mathbf{m}$ with treewidth w take time $O(|\mathbf{X}| \cdot 2^w)$ *(Dechter 1998; Koller et al. 2009).*

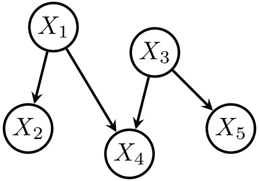
$\Longrightarrow$ *what about bounding the treewidth by design?*

Low-treewidth PGMs



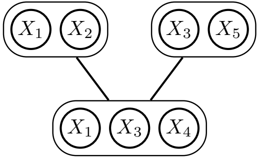
Trees

(Meilă et al. 2000)



Polytrees

(Dasgupta 1999)



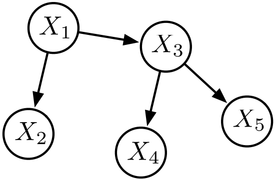
Thin Junction trees

(Bach et al. 2001)

If treewidth is bounded (e.g. $\approxeq 20$), exact MAR and CON inference is possible in practice

Tree distributions

A ***tree-structured BN*** *(Meilă et al. 2000)* where each $X_i \in \mathbf{X}$ has *at most* one parent Pa_{X_i}.

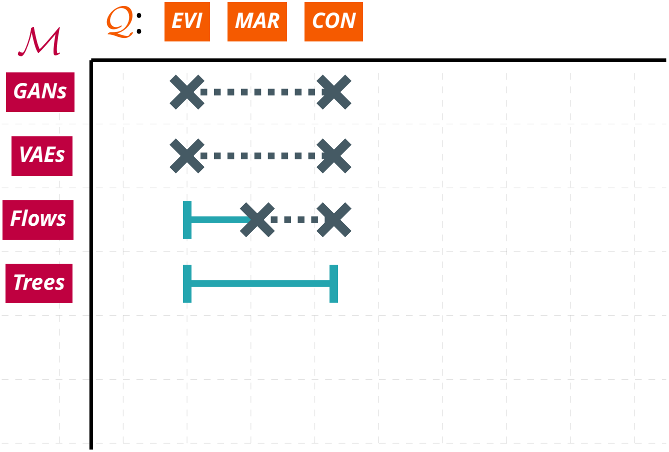


$$p(\mathbf{X}) = \prod_{i=1}^{n} p(x_i | \mathrm{Pa}_{x_i})$$

Exact querying: EVI, MAR, CON tasks *linear* for trees: $O(|\mathbf{X}|)$

Exact learning from d examples takes $O(|\mathbf{X}|^2 \cdot d)$ with the classical Chow-Liu algorithm[1]

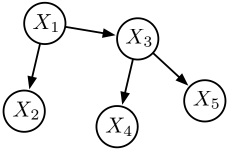
[1] *Chow et al., "Approximating discrete probability distributions with dependence trees", 1968*



tractable bands

What do we lose?

Expressiveness: Ability to represent rich and complex classes of distributions

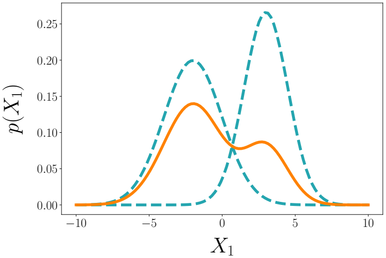


Bounded-treewidth PGMs lose the ability to represent *all possible distributions* ...

Cohen et al., "On the expressive power of deep learning: A tensor analysis", 2016
Martens and Medabalimi, "On the Expressive Efficiency of Sum Product Networks", 2014

Mixtures

Mixtures as a convex combination of k (simpler) probabilistic models

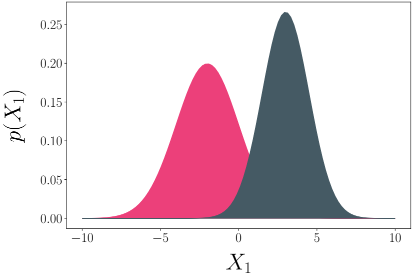


$$p(X) = w_1 \cdot p_1(X) + w_2 \cdot p_2(X)$$

EVI, MAR, CON queries scale linearly in k

Mixtures

Mixtures as a convex combination of k (simpler) probabilistic models



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

Mixtures are marginalizing a ***categorical latent variable*** Z with k values

$\Longrightarrow$ *increased expressiveness*

Expressiveness and efficiency

Expressiveness: Ability to represent rich and effective classes of functions

$\Longrightarrow$ *mixture of Gaussians can approximate any probability density!*

Cohen et al., "On the expressive power of deep learning: A tensor analysis", 2016
Martens and Medabalimi, "On the Expressive Efficiency of Sum Product Networks", 2014

Expressiveness and efficiency

Expressiveness: Ability to represent rich and effective classes of functions

$\Longrightarrow$ *mixture of Gaussians can approximate any probability density!*

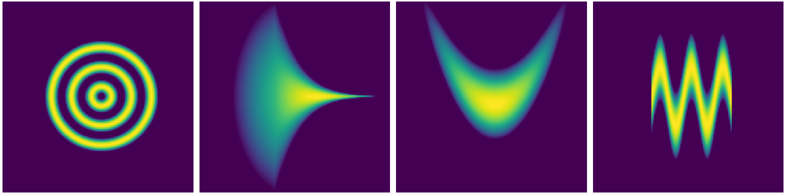
Expressive efficiency (aka Succinctness):

Ability to represent rich and effective classes of functions **compactly**

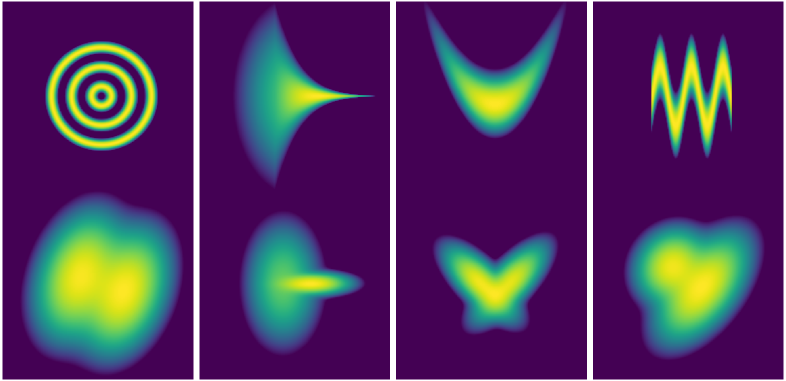
$\Longrightarrow$ *but how many components does a Gaussian mixture need?*

Cohen et al., "On the expressive power of deep learning: A tensor analysis", 2016
Martens and Medabalimi, "On the Expressive Efficiency of Sum Product Networks", 2014

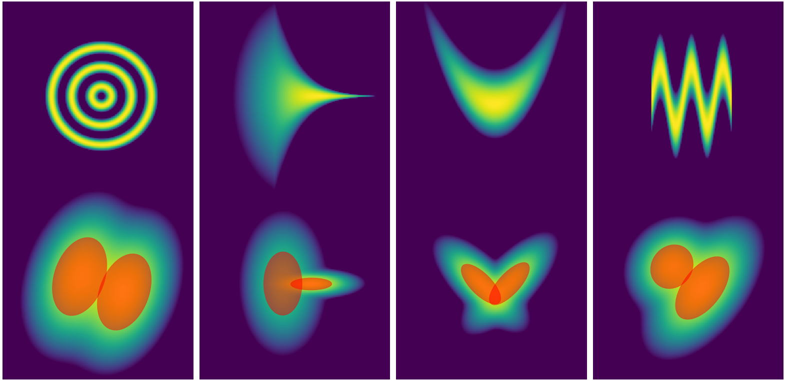
How expressive efficient are mixtures?



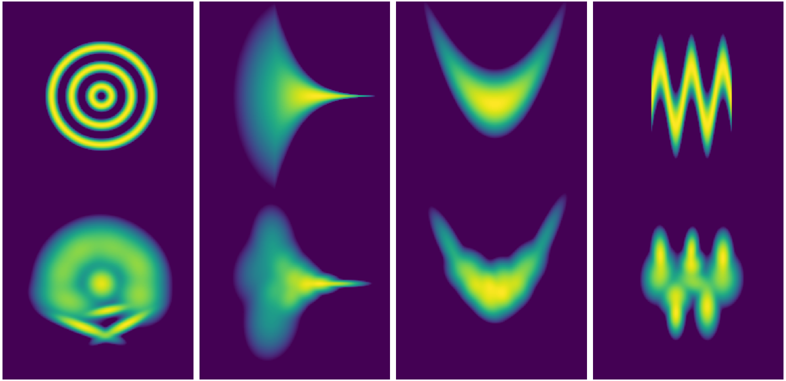
How expressive efficient are mixtures?



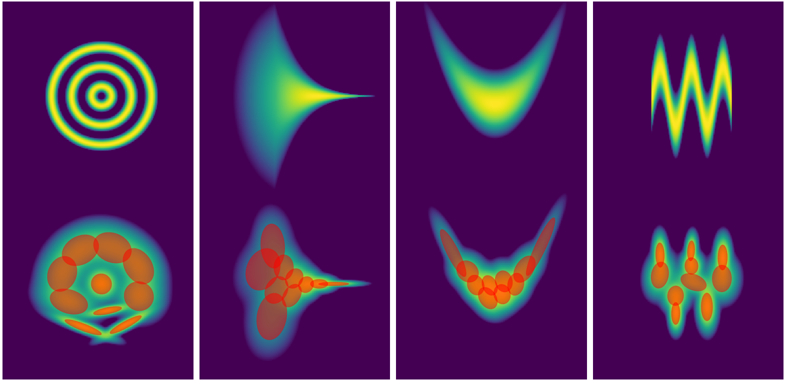
How expressive efficient are mixtures?



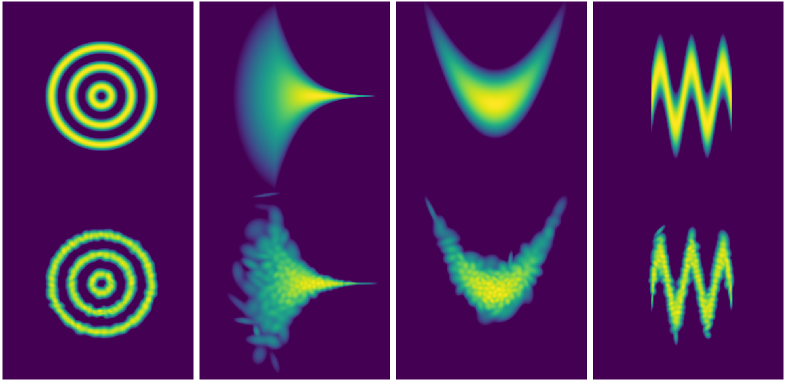
How expressive efficient are mixtures?



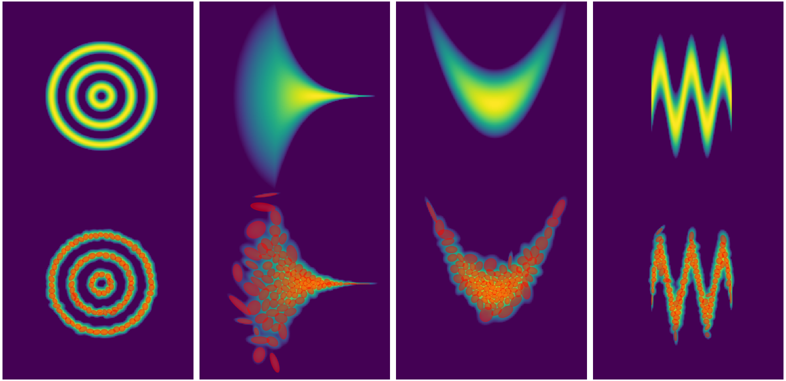
How expressive efficient are mixtures?



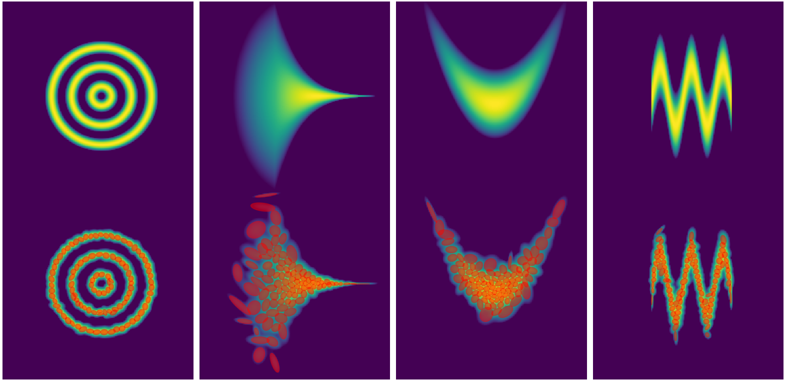
How expressive efficient are mixtures?



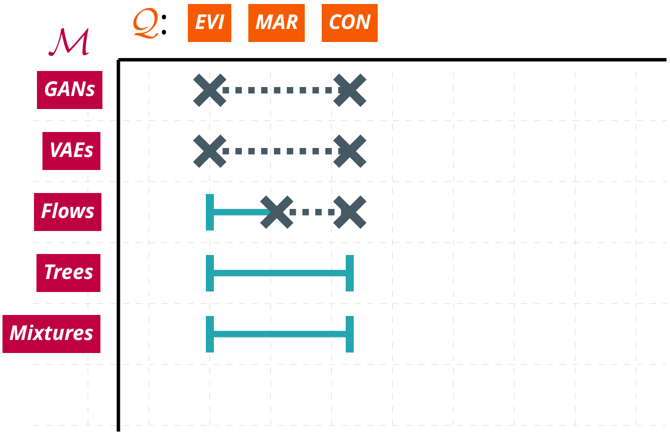
How expressive efficient are mixtures?



How expressive efficient are mixtures?



$\Rightarrow$ *solution: deep mixtures as in deep generative models*

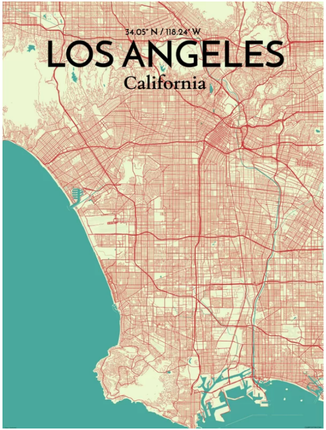


tractable bands

Maximum A Posteriori (MAP)

aka Most Probable Explanation (MPE)

q_5: *Which combination of roads is most likely to be jammed on Monday at 9am?*

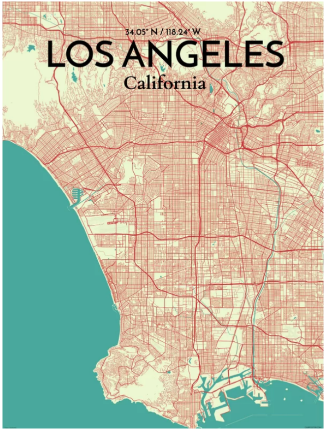


Maximum A Posteriori (MAP)

aka Most Probable Explanation (MPE)

$\mathbf{q}_5$: *Which combination of roads is most likely to be jammed on Monday at 9am?*

$$\mathbf{q}_5(\mathbf{m}) = \text{argmax}_{\mathbf{j}}\, p_{\mathbf{m}}(\mathbf{j}_1, \mathbf{j}_2, \ldots \mid \mathsf{Day}{=}\mathsf{M}, \mathsf{Time}{=}9)$$



© fineartamerica.com

Maximum A Posteriori (MAP)

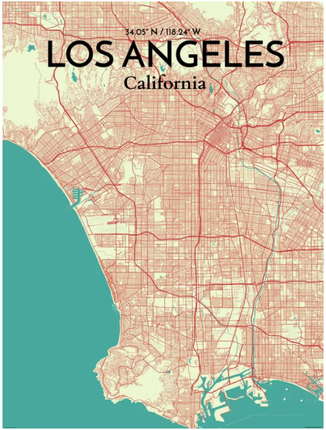
aka Most Probable Explanation (MPE)

q_5: *Which combination of roads is most likely to be jammed on Monday at 9am?*

$$q_5(\mathbf{m}) = \text{argmax}_{\mathbf{j}}\, p_{\mathbf{m}}(\mathbf{j}_1, \mathbf{j}_2, \ldots \mid \mathsf{Day}=\mathsf{M}, \mathsf{Time}=9)$$

General: $\text{argmax}_{\mathbf{q}}\ p_{\mathbf{m}}(\mathbf{q} \mid \mathbf{e})$

where $\mathbf{Q} \cup \mathbf{E} = \mathbf{X}$



© fineartamerica.com

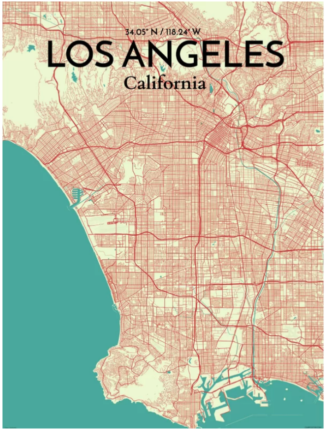
Maximum A Posteriori (MAP)

aka Most Probable Explanation (MPE)

$\mathbf{q}_5$: *Which combination of roads is most likely to be jammed on Monday at 9am?*

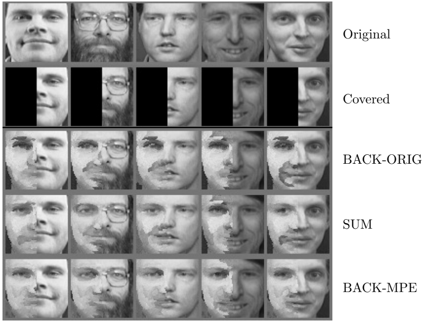
...***intractable*** for latent variable models!

$$\max_{\mathbf{q}} p_{\mathbf{m}}(\mathbf{q} \mid \mathbf{e}) = \max_{\mathbf{q}} \sum_{\mathbf{z}} p_{\mathbf{m}}(\mathbf{q}, \mathbf{z} \mid \mathbf{e})$$
$$\neq \sum_{\mathbf{z}} \max_{\mathbf{q}} p_{\mathbf{m}}(\mathbf{q}, \mathbf{z} \mid \mathbf{e})$$



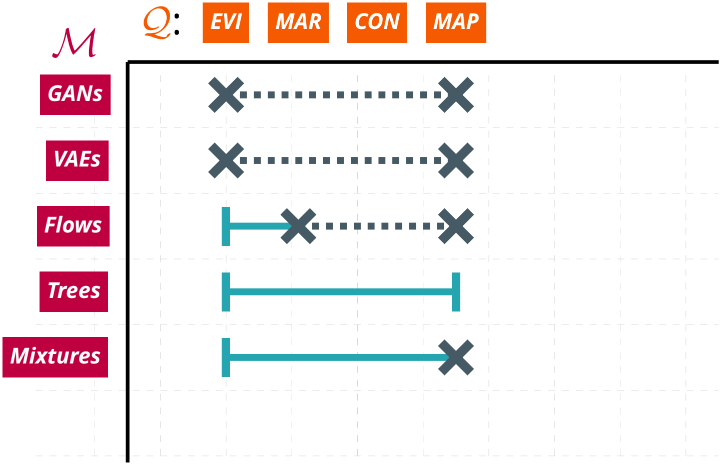
© fineartamerica.com

MAP inference : image inpainting



Predicting ***arbitrary patches***
given a ***single*** model
without the need of retraining.

Poon and Domingos, "Sum-Product Networks: a New Deep Architecture", 2011
Sguerra and Cozman, "Image classification using sum-product networks for autonomous flight of micro aerial vehicles", 2016

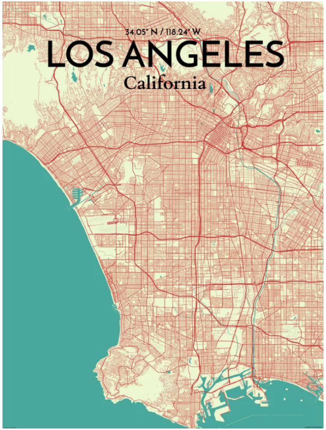


tractable bands

Marginal MAP (MMAP)

aka Bayesian Network MAP

q_6: *Which combination of roads is most likely to be jammed ~~on Monday~~ at 9am?*



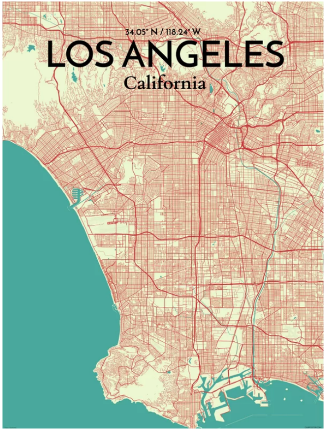
© fineartamerica.com

Marginal MAP (MMAP)

aka Bayesian Network MAP

$\mathbf{q}_6$: *Which combination of roads is most likely to be jammed ~~on Monday~~ at 9am?*

$$\mathbf{q}_6(\mathbf{m}) = \operatorname{argmax}_{\mathbf{j}}\, p_{\mathbf{m}}(\mathbf{j}_1, \mathbf{j}_2, \ldots \mid \mathsf{Time}\!=\!9)$$



Marginal MAP (MMAP)

aka Bayesian Network MAP

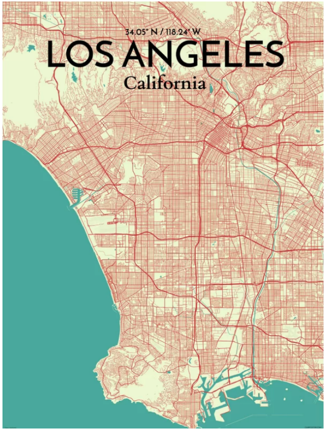
$\mathbf{q}_6$: *Which combination of roads is most likely to be jammed ~~on Monday~~ at 9am?*

$$\mathbf{q}_6(\mathbf{m}) = \text{argmax}_{\mathbf{j}}\ p_{\mathbf{m}}(\mathbf{j}_1, \mathbf{j}_2, \ldots \mid \mathsf{Time}{=}9)$$

General: $\text{argmax}_{\mathbf{q}}\ p_{\mathbf{m}}(\mathbf{q} \mid \mathbf{e})$

$$= \text{argmax}_{\mathbf{q}} \sum_{\mathbf{h}} p_{\mathbf{m}}(\mathbf{q}, \mathbf{h} \mid \mathbf{e})$$

where $\mathbf{Q} \cup \mathbf{H} \cup \mathbf{E} = \mathbf{X}$



Marginal MAP (MMAP)

aka Bayesian Network MAP

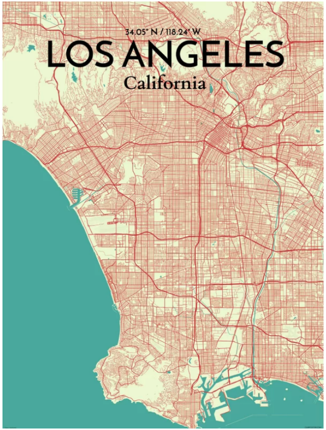
$\mathbf{q}_6$: *Which combination of roads is most likely to be jammed ~~on Monday~~ at 9am?*

$$\mathbf{q}_6(\mathbf{m}) = \operatorname{argmax}_{\mathbf{j}}\ p_{\mathbf{m}}(\mathbf{j}_1, \mathbf{j}_2, \ldots \mid \mathsf{Time}=9)$$

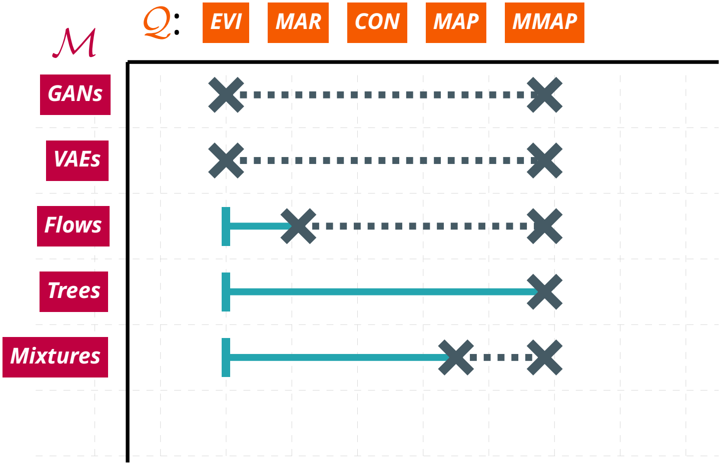
$\Longrightarrow$ *NP^{PP}-complete* (Park et al. 2006)

$\Longrightarrow$ *NP-hard for trees* (de Campos 2011)

$\Longrightarrow$ *NP-hard even for Naive Bayes* (ibid.)



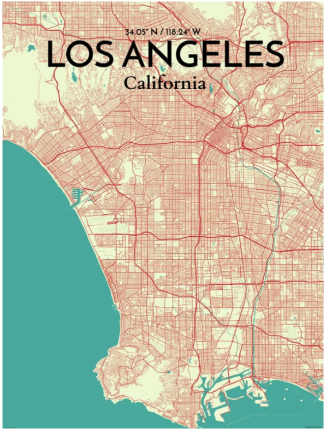
© fineartamerica.com



tractable bands

Advanced queries

q_2: *Which day is most likely to have a traffic jam on my route to campus?*



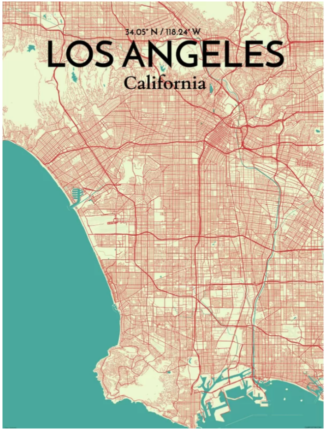
© fineartamerica.com

Advanced queries

$\mathbf{q}_2$: *Which day is most likely to have a traffic jam on my route to campus?*

$$\mathbf{q}_2(\mathbf{m}) = \operatorname{argmax}_{\mathsf{d}}\, p_{\mathbf{m}}(\mathsf{Day} = \mathsf{d} \land \bigvee_{i \in \mathsf{route}} \mathsf{Jam}_{\mathsf{Str}\ i})$$

$\Longrightarrow$ ***marginals + MAP + logical events***

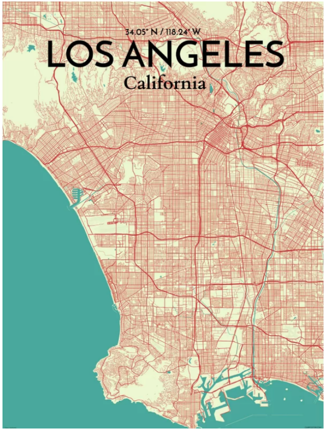


© fineartamerica.com

Advanced queries

q_2: *Which day is most likely to have a traffic jam on my route to campus?*

q_7: *What is the probability of seeing more traffic jams in Westwood than Hollywood?*



© fineartamerica.com

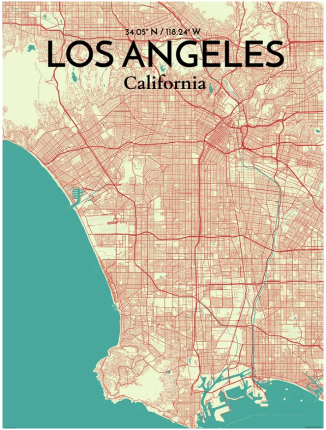
Bekker et al., "Tractable Learning for Complex Probability Queries", 2015

Advanced queries

q_2: *Which day is most likely to have a traffic jam on my route to campus?*

q_7: *What is the probability of seeing more traffic jams in Westwood than Hollywood?*

$\Rightarrow$ **counts + group comparison**



© fineartamerica.com

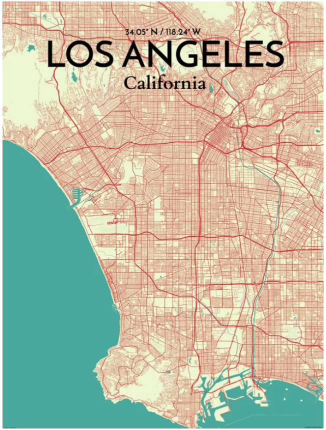
Bekker et al., "Tractable Learning for Complex Probability Queries", 2015

Advanced queries

q_2: *Which day is most likely to have a traffic jam on my route to campus?*

q_7: *What is the probability of seeing more traffic jams in Westwood than Hollywood?*

q_8: *Is traffic more uncertain on weekdays?*



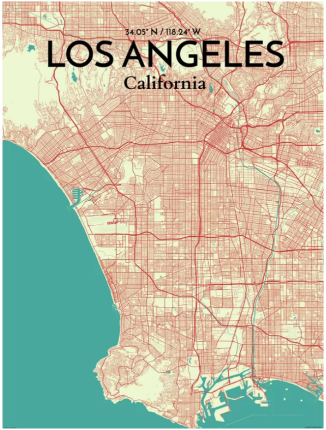
Advanced queries

q_2: *Which day is most likely to have a traffic jam on my route to campus?*

q_7: *What is the probability of seeing more traffic jams in Westwood than Hollywood?*

q_8: *Is traffic more uncertain on weekdays?*

$\Rightarrow$ ***information-theoretic queries***



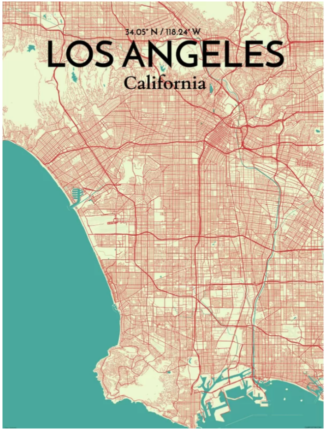
Advanced queries

q_2: *Which day is most likely to have a traffic jam on my route to campus?*

q_7: *What is the probability of seeing more traffic jams in Westwood than Hollywood?*

q_8: *Is traffic more uncertain on weekdays?*

q_9: *What is the causal effect of doing road works?*



Advanced queries

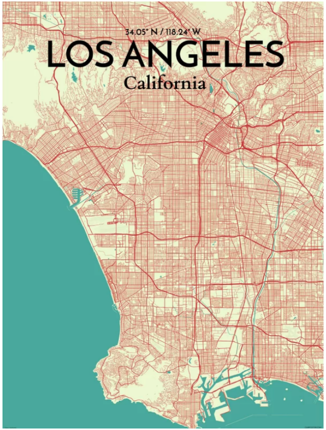
q_2: *Which day is most likely to have a traffic jam on my route to campus?*

q_7: *What is the probability of seeing more traffic jams in Westwood than Hollywood?*

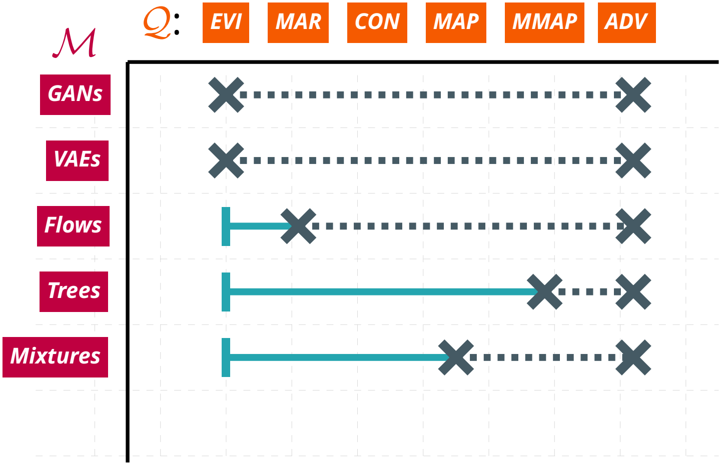
q_8: *Is traffic more uncertain on weekdays?*

q_9: *What is the causal effect of doing road works?*

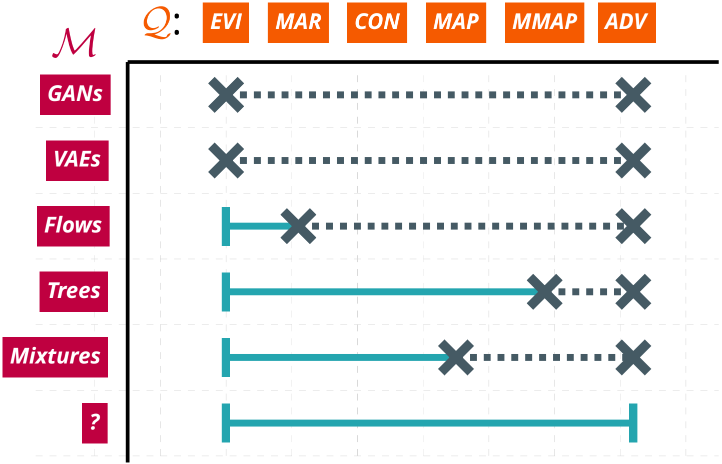
$\Longrightarrow$ ***causal backdoor estimation***



© fineartamerica.com



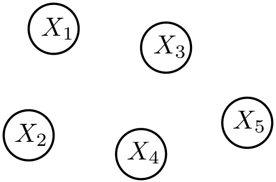
tractable bands



tractable bands

Fully factorized models

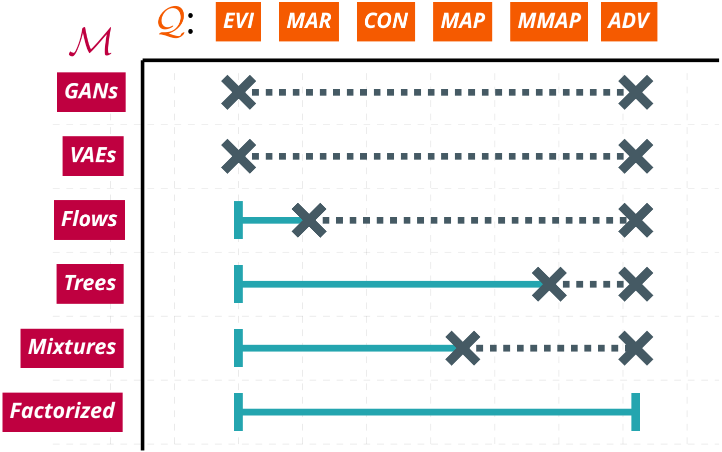
A completely disconnected graph. Example: Product of Bernoullis (PoBs)



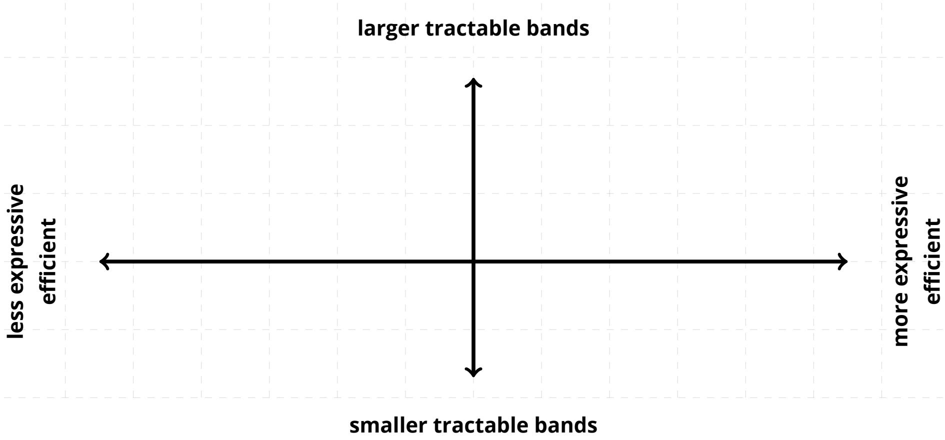
$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i)$$

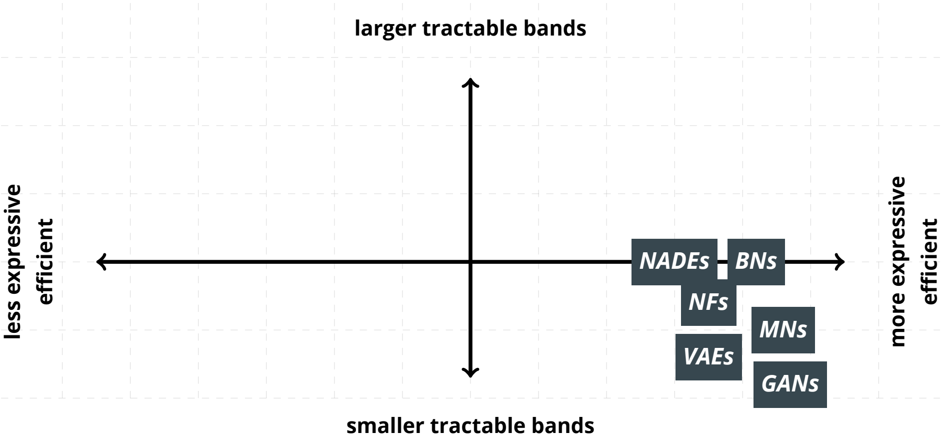
Complete evidence, marginals and MAP, MMAP inference is ***linear***!

$\Longrightarrow$ *but definitely not expressive...*

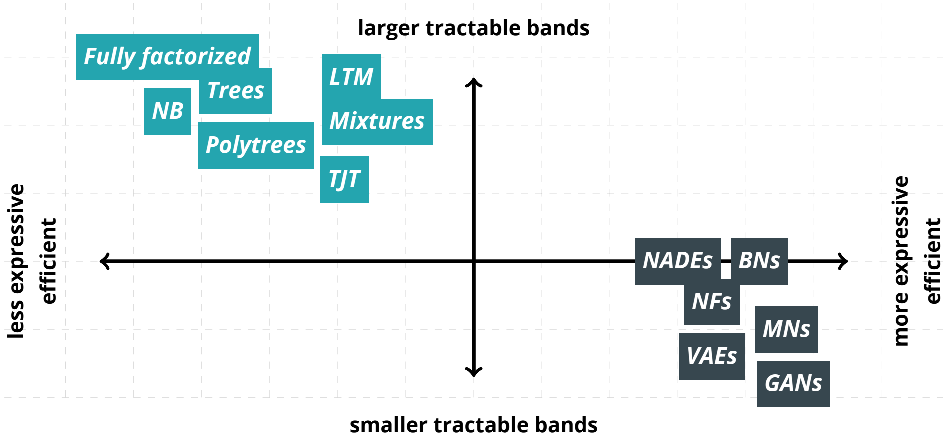


tractable bands

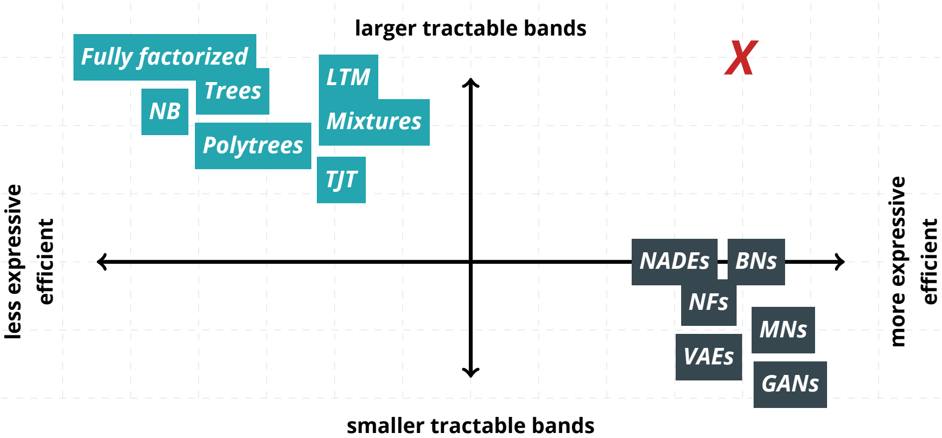

larger tractable bands
less expressive
efficient
more expressive
efficient
smaller tractable bands



Expressive models are not very tractable...



and tractable ones are not very expressive...



probabilistic circuits are at the "sweet spot"

Questions answered today

1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. ***What are probabilistic circuits and why are they tractable?*** (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. *Are all tractable distributions probabilistic circuits?* (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

Questions answered today

1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. ***What are probabilistic circuits and why are they tractable?*** (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. *Are all tractable distributions probabilistic circuits?* (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

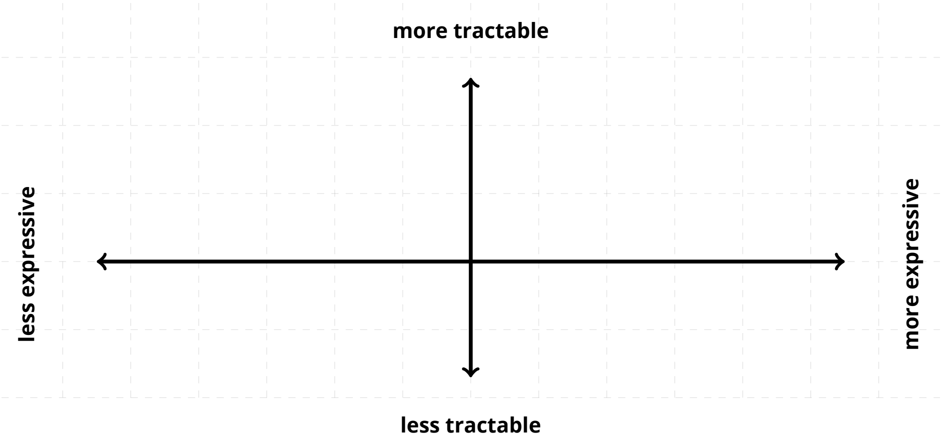
Probabilistic Circuits

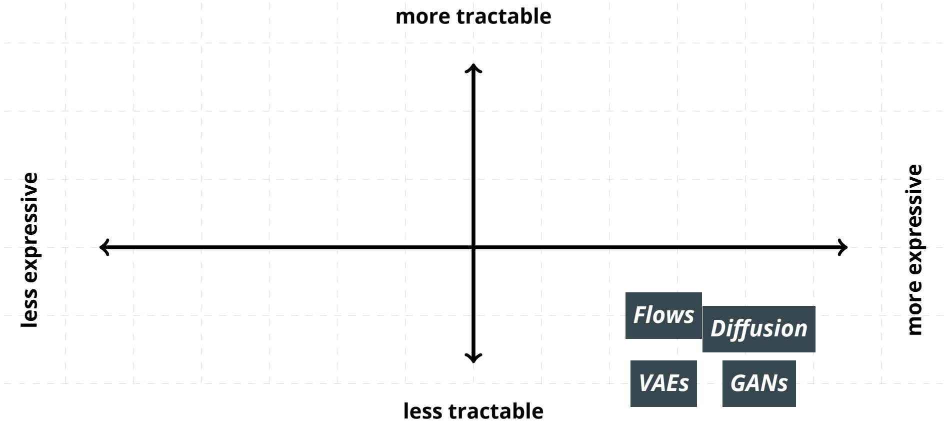
Goal

Given a reasoning task can we design a class of expressive models that is tractable for it?

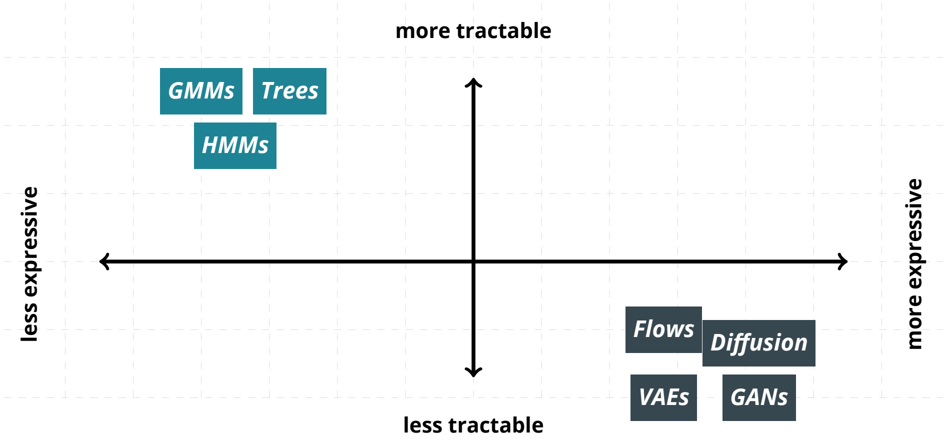
Goal

Given a reasoning task
can we design
a class of deep computational graphs
that is tractable for it?

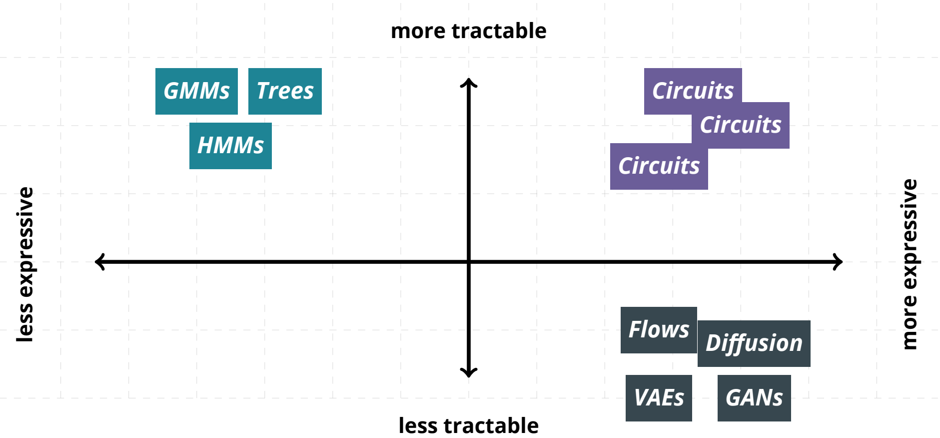

more tractable
less expressive
more expressive
less tractable



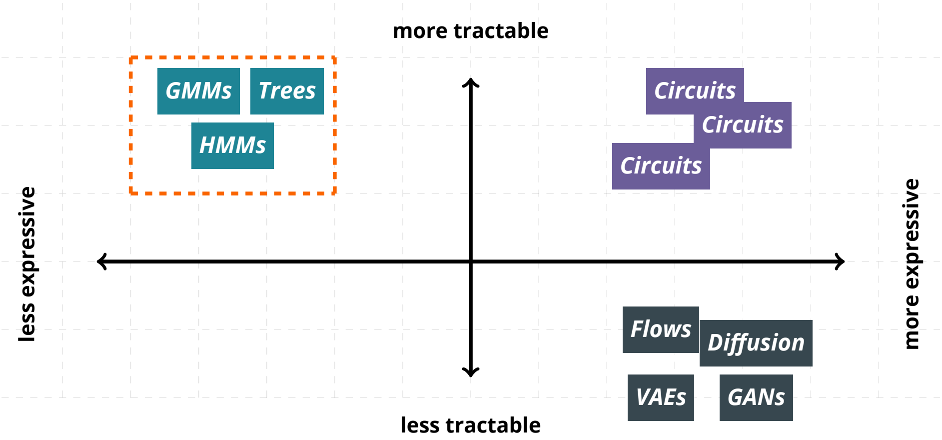
Expressive models are not very tractable…



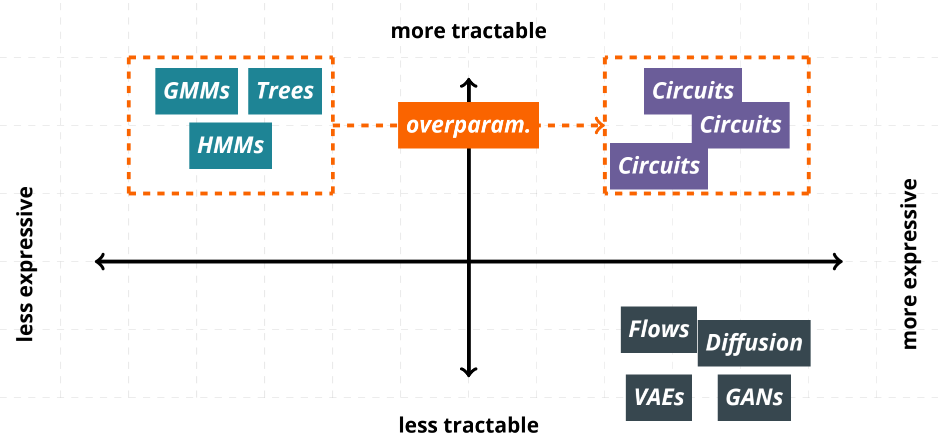
Tractable models are not that expressive...



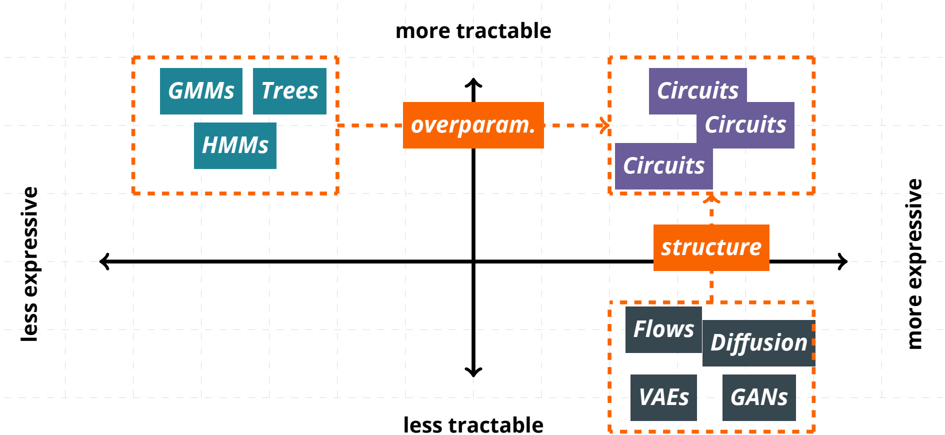
Circuits can be both expressive and tractable!



Start simple...



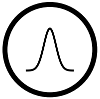
then make it more expressive!



impose structure!

Input distributions

as computational nodes



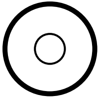
X

Base case: a single node encoding a distribution

$\Rightarrow$ *e.g., Gaussian PDF continuous random variable*

Input distributions

as computational nodes



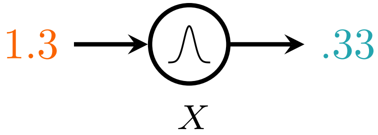
$\neg X$

Base case: a single node encoding a distribution

$\Rightarrow$ *e.g., indicators for X or $\neg X$ for Boolean random variable*

Input distributions

as computational nodes

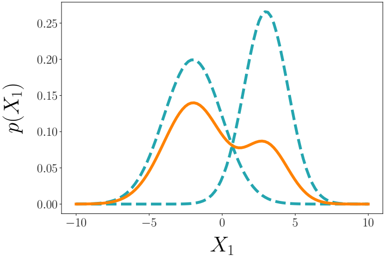


Simple distributions are tractable "black boxes" for:

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

Mixture models

as computational graphs

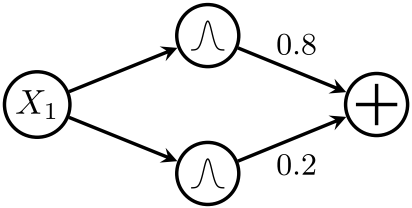


$$p(X) = w_1 \cdot p_1(X_1) + w_2 \cdot p_2(X_1)$$

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

Mixture models

as computational graphs

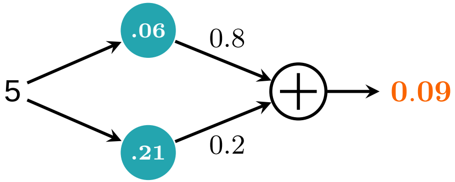


$$p(X_1) = 0.2 \cdot p_1(X_1) + 0.8 \cdot p_2(X_1)$$

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

Mixture models

as computational graphs

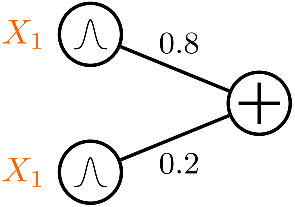


$$
\begin{aligned}
p(X=5) =&\, 0.2 \cdot p_1(X_1=5) \\
&+ 0.8 \cdot p_2(X_1=5)
\end{aligned}
$$

$\Longrightarrow$ *inference = feedforward evaluation*

Mixture models

as computational graphs



A simplified notation:

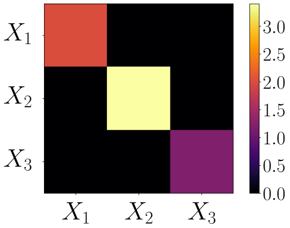
$\Rightarrow$ ***scopes*** *attached to inputs*

$\Rightarrow$ *edge directions omitted*

Factorizations

as computational graphs

$$p(X_1, X_2, X_3) = p(X_1) \cdot p(X_2) \cdot p(X_3)$$

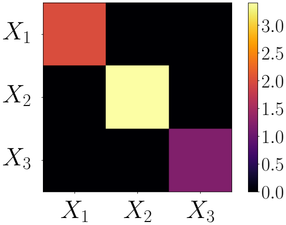


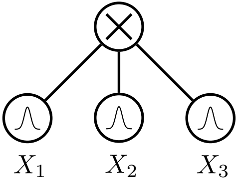
$\Longrightarrow$ *e.g. modeling a multivariate Gaussian with diagonal covariance matrix...*

Factorizations

as computational graphs

$$p(X_1, X_2, X_3) = p(X_1) \cdot p(X_2) \cdot p(X_3)$$



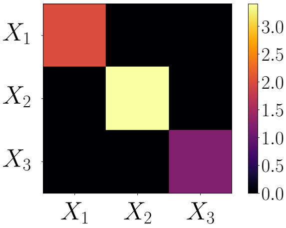


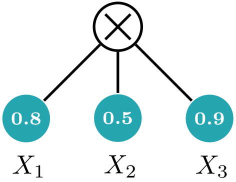
$\Rightarrow$ *...with a product node over some univariate Gaussian distribution*

Factorizations

as computational graphs

$$p(x_1, x_2, x_3) = p(x_1) \cdot p(x_2) \cdot p(x_3)$$



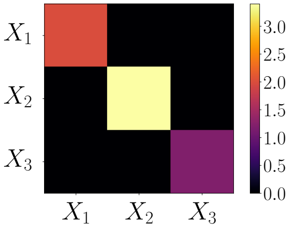


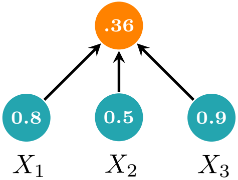
$\Longrightarrow$ *feedforward evaluation*

Factorizations

as computational graphs

$$p(x_1, x_2, x_3) = p(x_1) \cdot p(x_2) \cdot p(x_3)$$





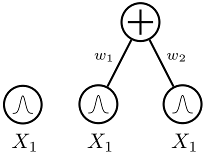
$\Longrightarrow$ *feedforward evaluation*

A grammar for tractable models

Recursive semantics of probabilistic circuits

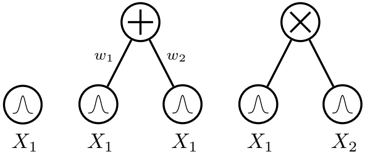
A grammar for tractable models

Recursive semantics of probabilistic circuits



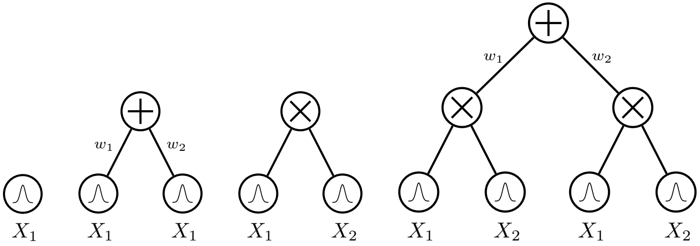
A grammar for tractable models

Recursive semantics of probabilistic circuits



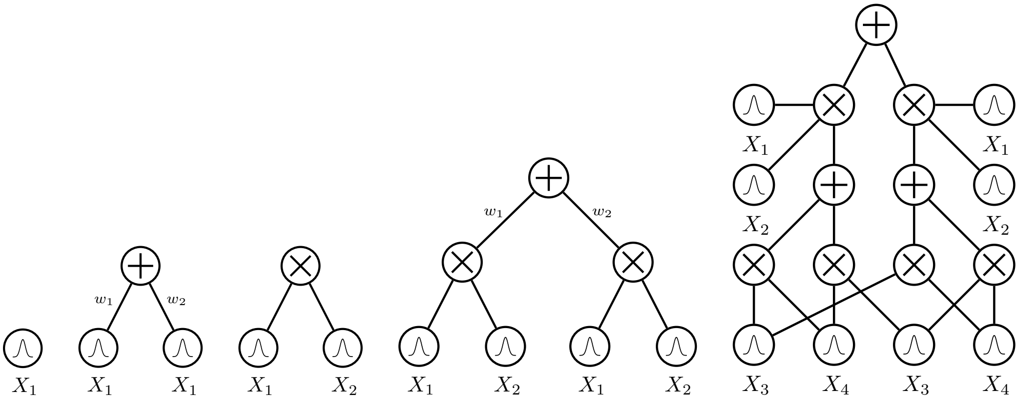
A grammar for tractable models

Recursive semantics of probabilistic circuits



A grammar for tractable models

Recursive semantics of probabilistic circuits



Building PCs in Python with SPFlow

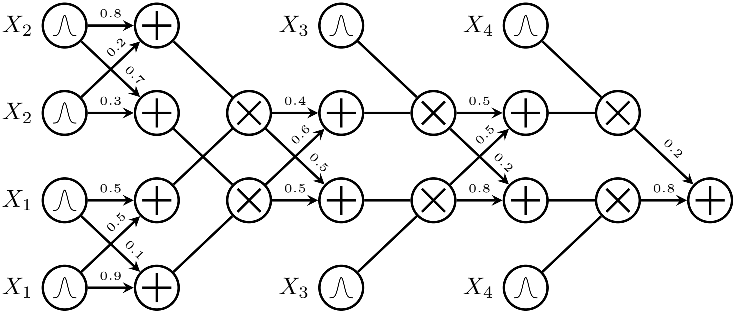
```
import spn.structure.leaves.parametric.Parametric as param
from param import Categorical, Gaussian

PC = 0.4 * (Categorical(p=[0.2, 0.8], scope=0) *
             (0.3 * (Gaussian(mean=1.0, stdev=1.0, scope=1) *
                     Categorical(p=[0.4, 0.6], scope=2))
            + 0.7 * (Gaussian(mean=-1.0, stdev=1.0, scope=1) *
                     Categorical(p=[0.6, 0.4], scope=2)))) \
    + 0.6 * (Categorical(p=[0.2, 0.8], scope=0) *
             Gaussian(mean=0.0, stdev=0.1, scope=1) *
             Categorical(p=[0.4, 0.6], scope=2))
```

Molina et al., "SPFlow: An easy and extensible library for deep probabilistic learning using sum-product networks", 2019

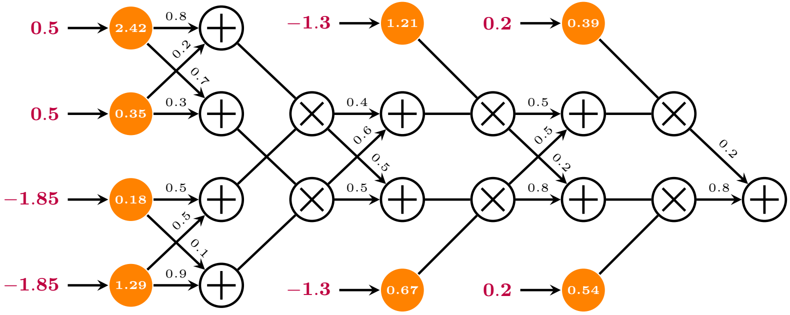
EVI queries = feedforward evaluation

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



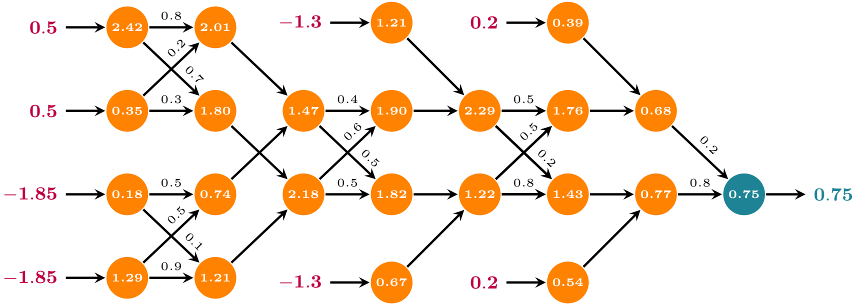
EVI queries = feedforward evaluation

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

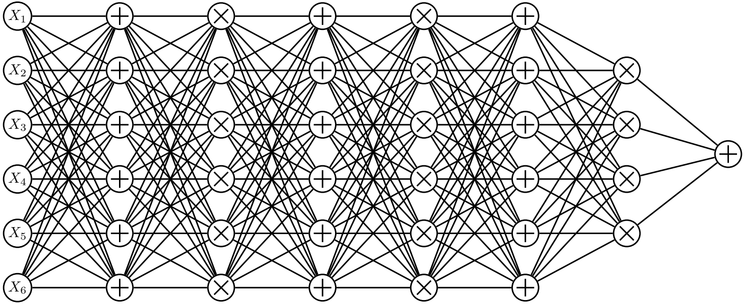


EVI queries = *feedforward* evaluation

$$p(X_1 = -1.85, X_2 = 0.5, X_3 = -1.3, X_4 = 0.2) = 0.75$$

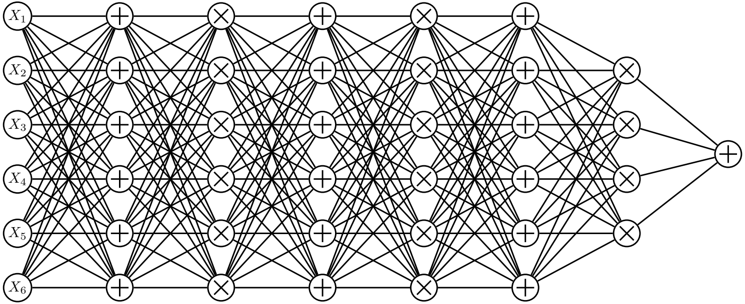


Just sum, products and distributions?



just arbitrarily compose them like a neural network!

Just sum, products and distributions?



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

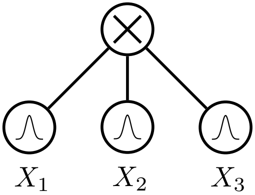
$\Rightarrow$ *structural properties needed for tractability*

Which structural constraints ensure tractability?

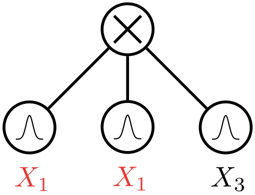
Decomposability

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

$\Longrightarrow$ *just like in factorization!*



decomposable circuit



non-decomposable circuit

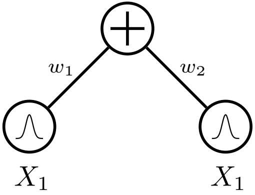
Darwiche and Marquis, "A knowledge compilation map", 2002

Smoothness

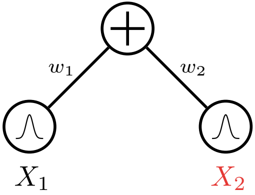
aka completeness

A sum node is smooth if its children depend of the same variable sets

$\Longrightarrow$ *otherwise not accounting for some variables*



smooth circuit



***non*-smooth** circuit

$\Longrightarrow$ *smoothness can be easily enforced (Shih et al. 2019)*

Darwiche and Marquis, "A knowledge compilation map", 2002

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

Computing arbitrary integrations (or summations)

$\Longrightarrow$ *linear in circuit size!*

E.g., suppose we want to compute Z
(the distribution's normalizing constant):

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

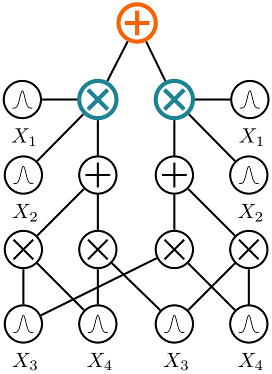
Smoothness + decomposability = tractable MAR

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*



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

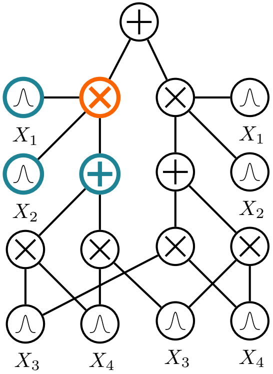
If $p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{x})p(\mathbf{y})p(\mathbf{z})$, (***decomposability***):

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

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

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

$\Longrightarrow$ *integrals decompose into easier ones*



Smoothness + decomposability = tractable MAR

Forward pass evaluation for MAR

$\Longrightarrow$ *linear in circuit size!*

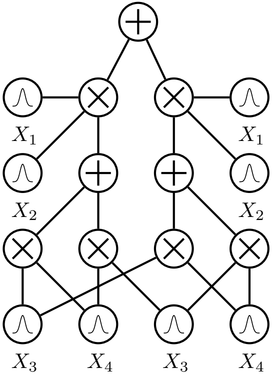
E.g. to compute $p(x_2, x_4)$:

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

  $\Longrightarrow$ *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

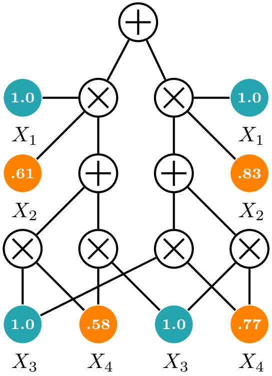
$\Longrightarrow$ *linear in circuit size!*

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

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

  $\Longrightarrow$ *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

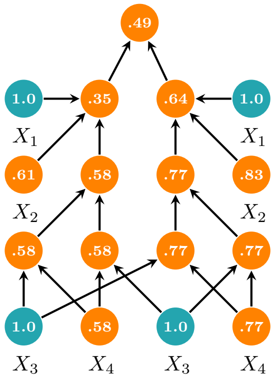
$\Rightarrow$ *linear in circuit size!*

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

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

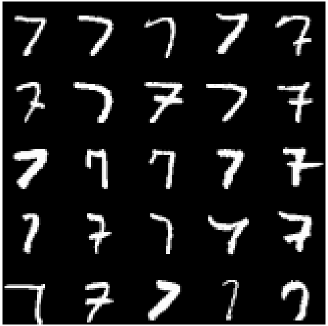
  $\Rightarrow$ *for normalized leaf distributions:* 1.0

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

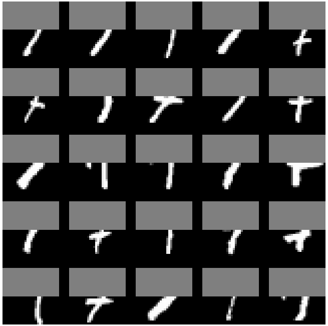


Tractable MAR

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

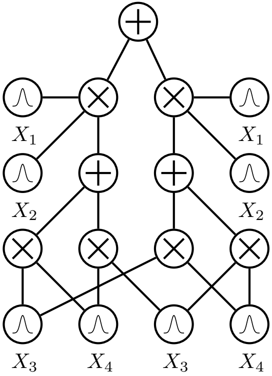
Smoothness + decomposability = tractable CON

Analogously, for arbitrary conditional queries:

$$p(\mathbf{q} \mid \mathbf{e}) = \frac{p(\mathbf{q}, \mathbf{e})}{p(\mathbf{e})}$$

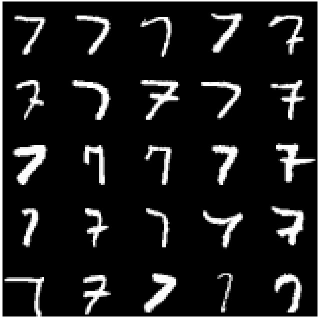
1. evaluate $p(\mathbf{q}, \mathbf{e})$ $\Longrightarrow$ *one feedforward pass*
2. evaluate $p(\mathbf{e})$ $\Longrightarrow$ *another feedforward pass*

$\Longrightarrow$ *...still linear in circuit size!*

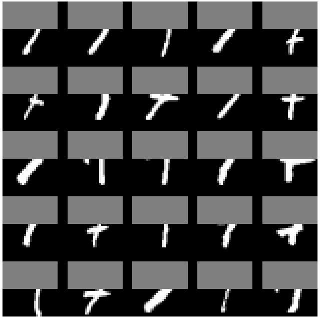


Tractable CON

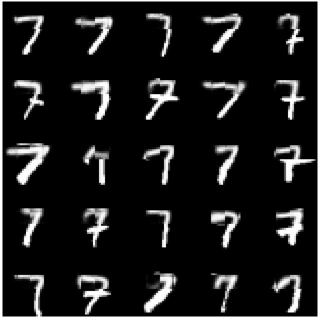
Original



Missing



Conditional sample



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

Generative models are still hard to control

more reasoning! | Generate image

Stable Diffusion

Generate a sentence using "frisbee", "caught" and "dog", following the given order.

Generate a sentence using "frisbee", "caught" and "dog", following the given order.

After a perfect throw, the frisbee glided through the air, and the dog, with incredible agility, caught it mid-flight.

ChatGPT

Generate a sentence using "frisbee", "caught" and "dog", following the given order.

After a perfect throw, the frisbee glided through the air, and the dog, with incredible agility, caught it mid-flight.

ChatGPT

That's not correct. Generate a sentence using "frisbee", "caught" and "dog". The keywords should appear in the order as specified.

Here's the correct sentence: The dog caught the frisbee in mid-air, showing off its amazing catching skills.

ChatGPT

Generate a sentence using "frisbee", "caught" and "dog", following the given order.

After a perfect throw, the frisbee glided through the air, and the dog, with incredible agility, caught it mid-flight.

ChatGPT

That's not correct. Generate a sentence using "frisbee", "caught" and "dog". The keywords should appear in the order as specified.

Here's the correct sentence: The dog caught the frisbee in mid-air, showing off its amazing catching skills.

ChatGPT

A frisbee is caught by a dog.
A pair of frisbee players are caught in a dog fight.

GeLaTo

What do we have?

Prefix: "The weather is"

Constraint α: text contains "winter"

Model only does $p(\text{next-token}|\text{prefix}) =$

cold	0.05
warm	0.10

What do we need?

Prefix: “The weather is”

Constraint α: text contains “winter”

Generate from $p(\text{next-token}|\text{prefix}, \alpha) =$

cold	0.50
warm	0.01

$$\propto \sum_{\text{text}} p(\text{next-token}, \text{text}, \text{prefix}, \alpha)$$

Marginalization!

Computing $p(\alpha \mid x_{1:t+1})$

For **α** in conjunctive normal form (CNF):

$$(w_{1,1} \lor \ldots \lor w_{1,d1}) \land \ldots \land (w_{m,1} \lor \ldots \lor w_{m,dm})$$

where each w_{ij} is a keyword (i.e. a **string of tokens**),
representing the constraint that w_{ij} appears in the generated text.

e.g., **α** = ("swims" ∨ "like swimming") ∧ ("lake" ∨ "pool")

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. [Tractable Control for Autoregressive Language Generation](), 2023.

Computing $p(\alpha \mid x_{1:t+1})$

For **α** in conjunctive normal form (CNF):

$$(w_{1,1} \lor \ldots \lor w_{1,d1}) \land \ldots \land (w_{m,1} \lor \ldots \lor w_{m,dm})$$

where each w_{ij} is a keyword (i.e. a **string of tokens**),
representing the constraint that w_{ij} appears in the generated text.

e.g., **α** = ("swims" ∨ "like swimming") ∧ ("lake" ∨ "pool")

Efficient algorithm:
For m clauses and sequence length n, time-complexity for generation is $O(2^{|m|}n)$
when p is a *hidden Markov model* (see general probabilistic circuit case later).

Trick: dynamic programming with clever preprocessing and local belief updates

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

CommonGen: a Challenging Benchmark

Given 3-5 concepts (keywords), our goal is to generate a sentence using all keywords, which can appear in any order and any form of inflections. e.g.,

Input: snow drive car

Reference 1: A car drives down a snow covered road.

Reference 2: Two cars drove through the snow.

$$(w_{1,1} \vee \ldots \vee w_{1,d1}) \wedge \ldots \wedge (w_{m,1} \vee \ldots \vee w_{m,dm})$$

Each clause represents the inflections for one keyword.

GeLaTo Overview

Lexical Constraint α: sentence contains keyword "winter"

Constrained Generation: $\Pr(x_{t+1} \mid \alpha, x_{1:t} = \text{"the weather is"})$

intractable

efficient

Pre-trained Language Model

Tractable Probabilistic Model

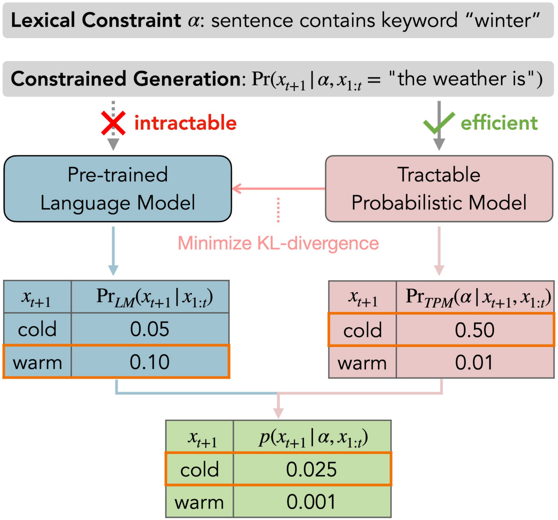
Minimize KL-divergence

x_{t+1}	$\Pr_{LM}(x_{t+1} \mid x_{1:t})$
cold	0.05
warm	$\boxed{0.10}$

x_{t+1}	$\Pr_{TPM}(\alpha \mid x_{t+1}, x_{1:t})$
cold	$\boxed{0.50}$
warm	0.01

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

GeLaTo Overview



Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. [Tractable Control for Autoregressive Language Generation](), 2023.

Step 2: Control p_{gpt} via p_{hmm}

Unsupervised

Language model is not
fine-tuned/prompted to satisfy constraints

By Bayes rule:

$$p_{gpt}(x_{t+1} \mid x_{1:t}, \alpha) \propto p_{gpt}(\alpha \mid x_{1:t+1}) \cdot p_{gpt}(x_{t+1} \mid x_{1:t})$$

Assume $p_{hmm}(\alpha \mid x_{1:t+1}) \approx p_{gpt}(\alpha \mid x_{1:t+1})$, we generate from:

$$p(x_{t+1} \mid x_{1:t}, \alpha) \propto p_{hmm}(\alpha \mid x_{1:t+1}) \cdot p_{gpt}(x_{t+1} \mid x_{1:t})$$

Method	*Generation Quality*								*Constraint Satisfaction*			
	ROUGE-L		BLEU-4		CIDEr		SPICE		Coverage		Success Rate	
Unsupervised	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*
InsNet (Lu et al., 2022a)	-	-	18.7	-	-	-	-	-	**100.0**	-	**100.0**	-
NeuroLogic (Lu et al., 2021)	-	41.9	-	24.7	-	14.4	-	27.5	-	96.7	-	-
A*esque (Lu et al., 2022b)	-	**44.3**	-	28.6	-	15.6	-	29.6	-	97.1	-	-
NADO (Meng et al., 2022)	-	-	26.2	-	-	-	-	-	96.1	-	-	-
GeLaTo	**44.6**	44.1	**29.9**	**29.4**	**16.0**	**15.8**	**31.3**	**31.0**	**100.0**	**100.0**	**100.0**	**100.0**

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

Step 2: Control p_{gpt} via p_{hmm}

Supervised

Language model is fine-tuned to perform constrained generation (e.g. seq2seq)

Empirically $p_{HMM}(\alpha \mid x_{1:t+1}) \approx p_{gpt}(\alpha \mid x_{1:t+1})$ does not hold well enough;

we view $p_{HMM}(x_{t+1} \mid x_{1:t}, \alpha)$ and $p_{gpt}(x_{t+1} \mid x_{1:t})$ as classifiers trained for the same task with different biases; thus we generate from their ***weighted geometric mean***:

$$p(x_{t+1} \mid x_{1:t}, \alpha) \propto p_{hmm}(x_{t+1} \mid x_{1:t}, \alpha)^{w} \cdot p_{gpt}(x_{t+1} \mid x_{1:t})^{1-w}$$

Method	*Generation Quality*								*Constraint Satisfaction*			
	ROUGE-L		BLEU-4		CIDEr		SPICE		Coverage		Success Rate	
Supervised	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*
NeuroLogic (Lu et al., 2021)	-	42.8	-	26.7	-	14.7	-	30.5	-	97.7	-	$93.9^{\dagger}$
A*esque (Lu et al., 2022b)	-	43.6	-	28.2	-	15.2	-	30.8	-	97.8	-	$97.9^{\dagger}$
NADO (Meng et al., 2022)	$44.4^{\dagger}$	-	30.8	-	$16.1^{\dagger}$	-	**32.0**†	-	97.1	-	$88.8^{\dagger}$	-
GeLaTo	**46.0**	**45.6**	**34.1**	**32.9**	**16.7**	**16.8**	31.3	**31.9**	**100.0**	**100.0**	**100.0**	**100.0**

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. [Tractable Control for Autoregressive Language Generation](), 2023.

Advantages of GeLaTo:

1. Constraint α is <u>guaranteed to be satisfied</u>: for any next-token x_{t+1} that would make α unsatisfiable, $p(x_{t+1} \mid x_{1:t}, \alpha) = 0$ for both settings.

2. Training p_{hmm} <u>does not depend on α</u>, which is only imposed at inference (generation) time. Once p_{hmm} is trained, we can impose whatever α.

3. We can impose <u>additional tractable constraints</u>:
    - The keywords are generated following a particular order.
    - (Some) keywords must appear at a particular position.
    - (Some) keywords must not appear in the generated sentence.

Conclusion: you can control an intractable generative model using a tractable probabilistic circuit.

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

We can also decompose bottom-up a MAP query:

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

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

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

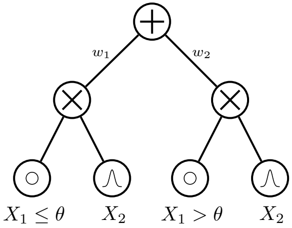
$\Longrightarrow$ *MAP for latent variable models is **intractable*** *(Conaty et al. 2017)*

Determinism

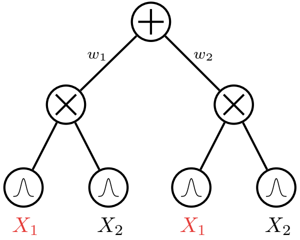
aka selectivity

A sum node is deterministic if the output of only one children is non zero for any input

$\Rightarrow$ *e.g. if their distributions have disjoint support*



deterministic circuit



non-deterministic circuit

Darwiche and Marquis, "A knowledge compilation map", 2002

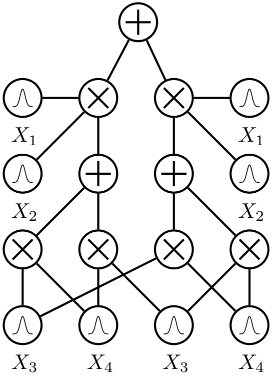
Determinism + decomposability = tractable MAP

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

$\Rightarrow$ *linear in circuit size!*

E.g., suppose we want to compute:

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

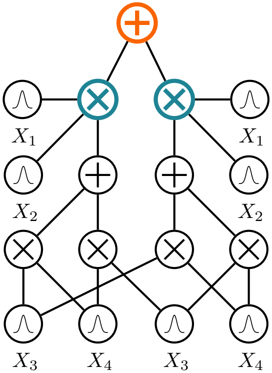


Determinism + decomposability = tractable MAP

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):

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

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

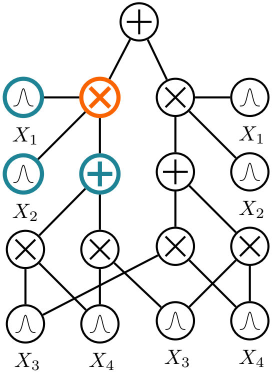


Determinism + decomposability = tractable MAP

If $p(\mathbf{q}, \mathbf{e}) = p(\mathbf{q_x}, \mathbf{e_x}, \mathbf{q_y}, \mathbf{e_y}) = p(\mathbf{q_x}, \mathbf{e_x})p(\mathbf{q_y}, \mathbf{e_y})$ (***decomposable*** product node):

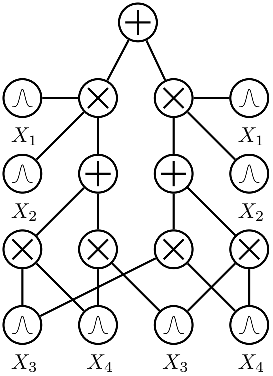
$$\max_{\mathbf{q}} p(\mathbf{q} \mid \mathbf{e}) = \max_{\mathbf{q}} p(\mathbf{q}, \mathbf{e})$$
$$= \max_{\mathbf{q_x}, \mathbf{q_y}} p(\mathbf{q_x}, \mathbf{e_x}, \mathbf{q_y}, \mathbf{e_y})$$
$$= \max_{\mathbf{q_x}} p(\mathbf{q_x}, \mathbf{e_x}) \cdot \max_{\mathbf{q_y}} p(\mathbf{q_y}, \mathbf{e_y})$$

$\Longrightarrow$ *solving optimization independently*



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

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

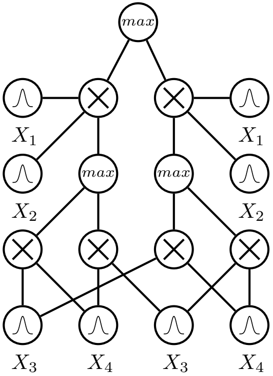


Determinism + decomposability = tractable MAP

Evaluating the circuit twice:
bottom-up and ***top-down*** $\Longrightarrow$ *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

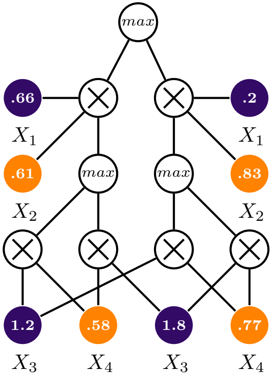


Determinism + decomposability = tractable MAP

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

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

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



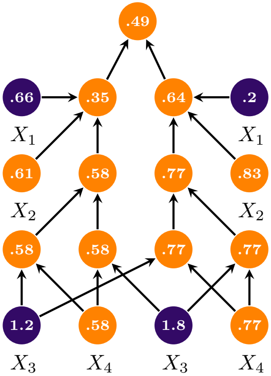
Determinism + **decomposability** = **tractable MAP**

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

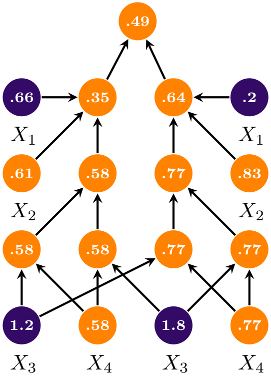


Determinism + decomposability = tractable MAP

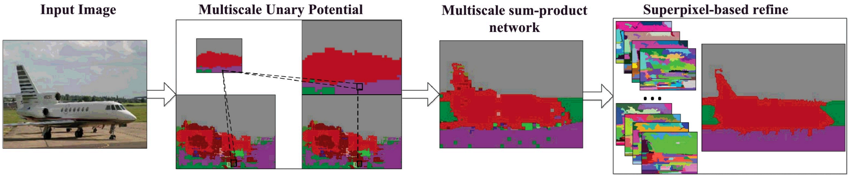
Evaluating the circuit twice:
bottom-up and ***top-down*** $\Rightarrow$ *still linear in circuit size!*

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



MAP inference : image segmentation



Semantic segmentation is MAP over joint pixel and label space

Even approximate MAP for non-deterministic circuits (SPNs) delivers good performances.

Rathke et al., "Locally adaptive probabilistic models for global segmentation of pathological oct scans", 2017
Yuan et al., "Modeling spatial layout for scene image understanding via a novel multiscale sum-product network", 2016
Friesen and Domingos, "Submodular Sum-product Networks for Scene Understanding", 2016

How expressive?

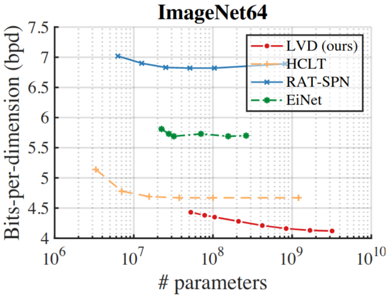
Dataset	Sparse PC (ours)	HCLT	RatSPN	IDF	BitSwap	BB-ANS	McBits
MNIST	**1.14**	1.20	1.67	1.90	1.27	1.39	1.98
EMNIST(MNIST)	**1.52**	1.77	2.56	2.07	1.88	2.04	2.19
EMNIST(Letters)	**1.58**	1.80	2.73	1.95	1.84	2.26	3.12
EMNIST(Balanced)	**1.60**	1.82	2.78	2.15	1.96	2.23	2.88
EMNIST(ByClass)	**1.54**	1.85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

competitive with Flows and VAEs!

Dang et al., "Sparse Probabilistic Circuits via Pruning and Growing", 2022

How scalable?

Dataset	TPMs				DGMs		
	LVD (ours)	HCLT	EiNet	RAT-SPN	Glow	RealNVP	BIVA
ImageNet32	**4.39**±0.01	4.82	5.63	6.90	4.09	4.28	3.96
ImageNet64	**4.12**±0.00	4.67	5.69	6.82	3.81	3.98	-
CIFAR	**4.38**±0.02	4.61	5.81	6.95	3.35	3.49	3.08



up to billions of parameters

Liu et al., "Scaling Up Probabilistic Circuits by Latent Variable Distillation", 2022

Questions answered today

1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. ***What is the connection to logical circuit languages?*** (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. *Are all tractable distributions probabilistic circuits?* (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

Questions answered today

1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. ***What is the connection to logical circuit languages?*** (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. *Are all tractable distributions probabilistic circuits?* (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

Logical Circuits

Tractability to other semi-rings

Tractable probabilistic inference exploits ***efficient summation for decomposable functions*** in the probability commutative semiring:

$$(\mathbb{R}, +, \times, 0, 1)$$

analogously efficient computations can be done in other semi-rings:

$$(\mathbb{S}, \oplus, \otimes, 0_{\oplus}, 1_{\otimes})$$

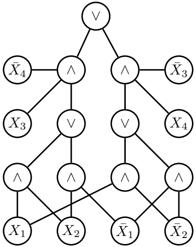
$\Longrightarrow$ *Algebraic model counting* (Kimmig et al. 2017), *Semi-ring programming* (Belle et al. 2016)

Historically, ***very well studied for boolean functions***:

$$(\mathbb{B} = \{0, 1\}, \vee, \wedge, 0, 1)$$

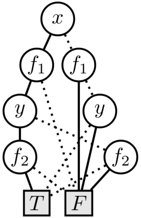
$\Longrightarrow$ *logical circuits!*

Logical circuits



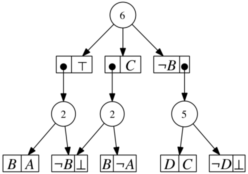
s/d-D/NNFs

(Darwiche et al. 2002a)



O/BDDs

(Bryant 1986)



SDDs

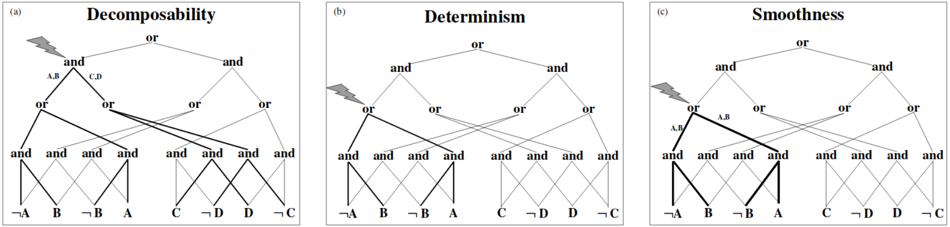
(Darwiche 2011a)

Logical circuits are compact representations for boolean functions...

Logical circuits

structural properties

...and like probabilitistic circuits, one can define ***structural properties***: (*structured*) *decomposability*, *smoothness*, *determinism* allowing for tractable computations

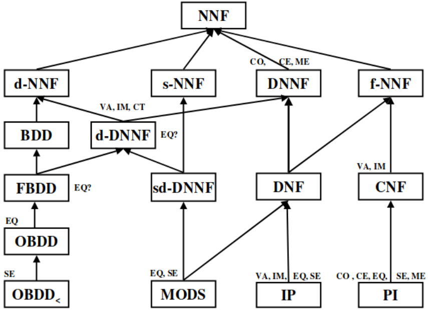


Darwiche and Marquis, "A knowledge compilation map", 2002

Logical circuits

a knowledge compilation map

...inducing ***a hierarchy of tractable logical circuit families***



Darwiche and Marquis, "A knowledge compilation map", 2002

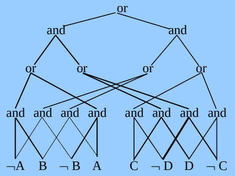
Knowledge Compilation

encoding

```
(A and (not B))
or(C and (not D))
or ((not C) and D)
...
```

Compiler

NNF Circuit

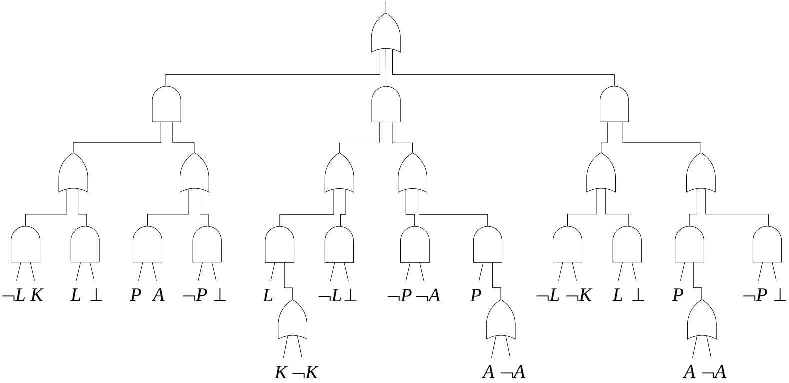


MAJ-MAJ-SAT
E-MAJ-SAT
MAJ-SAT
SAT

Answer in Linear Time

NNF Circuits

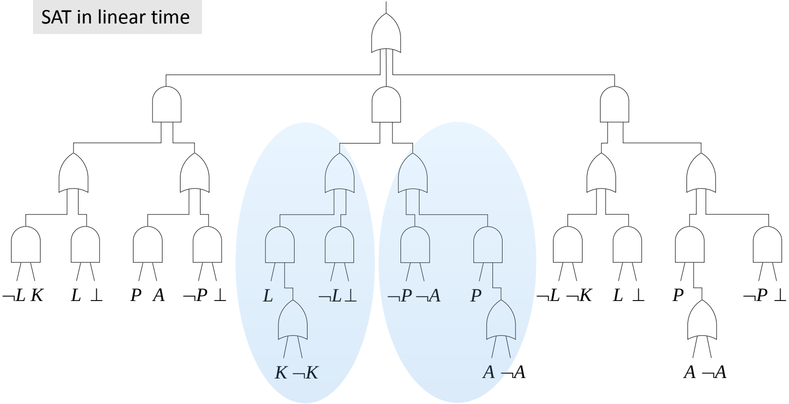
$P \vee L$
$A \Rightarrow P$
$K \Rightarrow (P \vee L)$



Decomposability (DNNF)

Darwiche, JACM 2001

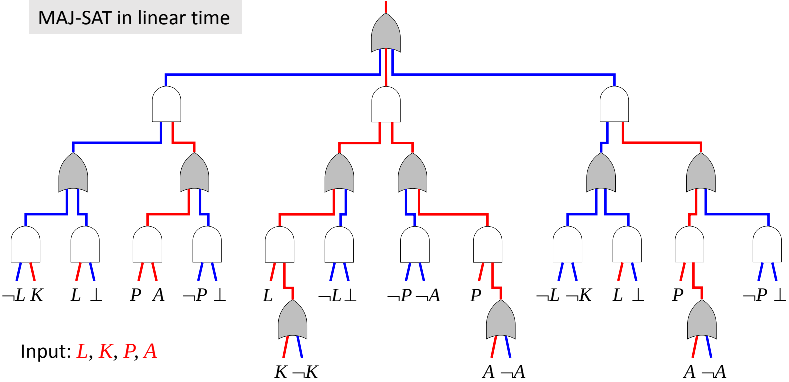
SAT in linear time



Determinism (d-DNNF)

Darwiche, JANCL 2000

MAJ-SAT in linear time



Input: L, K, P, A

Decomposability + ***determinism*** = ***tractable (W)MC***

Model counting problem: given a Boolean formula Δ, compute the number of satisfying assignments.

***Weighted model counting* (WMC):**

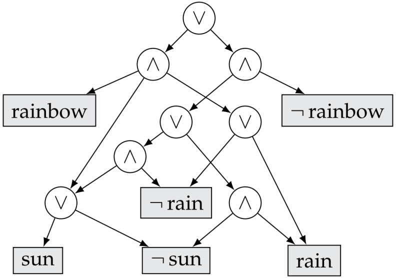
$$\mathrm{WMC}(\Delta, w) = \sum_{\mathbf{x} \models \Delta} \prod_{l \in \mathbf{x}} w(l)$$

$\Longrightarrow$ *linear in circuit size!*

Decomposability + determinism = tractable (W)MC

To compute $\mathrm{WMC}(\Delta, w)$:

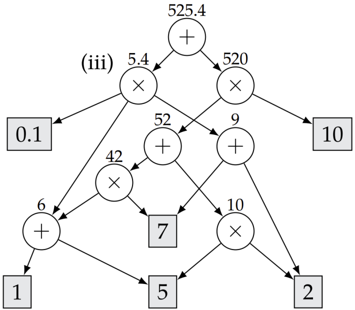
- Turn OR gates to sum nodes and AND gates to product nodes
- Replace each literal l with its weight $w(l)$
- bottom-up evaluation



Decomposability + **determinism** = **tractable (W)MC**

To compute $\mathrm{WMC}(\Delta, w)$:

- Turn OR gates to sum nodes and AND gates to product nodes
- Replace each literal l with its weight $w(l)$
- bottom-up evaluation



Probabilistic inference by WMC

connection to probabilistic circuits through WMC

1. Encode probabilistic model as WMC formula (Δ, w)
2. Compile Δ into a logical circuit (e.g. d-DNNF, OBDD, SDD, etc.)
3. Tractable MAR/CON by tractable WMC on circuit
4. Answer *complex queries* tractably by enforcing *more structural properties*!

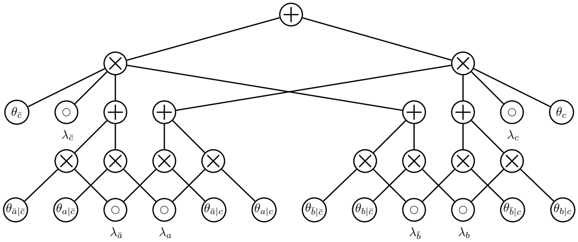
Chavira and Darwiche, "On probabilistic inference by weighted model counting", 2008

Probabilistic inference by WMC

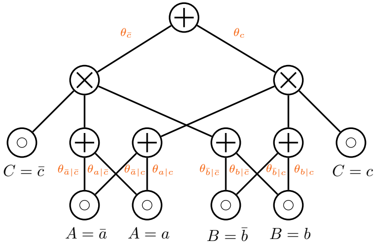
connection to probabilistic circuits through WMC

Resulting compiled WMC circuit **equivalent to probabilistic circuit**

$\Rightarrow$ *parameter variables* $\longrightarrow$ *edge parameters*



Compiled circuit of WMC encoding



Equivalent probabilistic circuit

Questions answered today

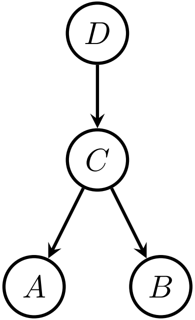
1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. ***How do I compile my favorite model into a circuit?*** (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. *Are all tractable distributions probabilistic circuits?* (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

Questions answered today

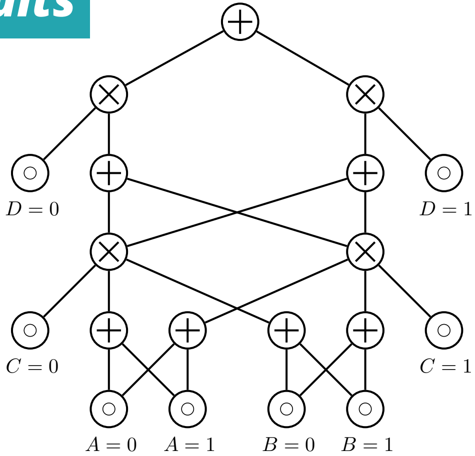
1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. ***How do I compile my favorite model into a circuit?*** (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. *Are all tractable distributions probabilistic circuits?* (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

From tree BN to circuits

via compilation

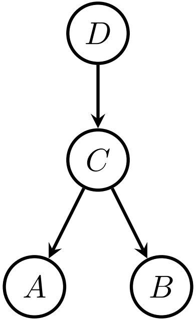


$\rightarrow$



From tree BN to circuits

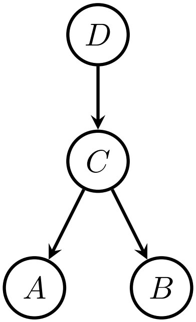
via compilation



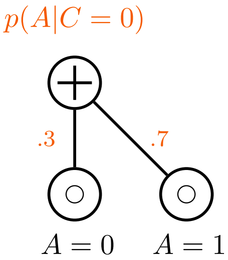
Bottom-up ***compilation***: starting from leaves...

From tree BN to circuits

via compilation

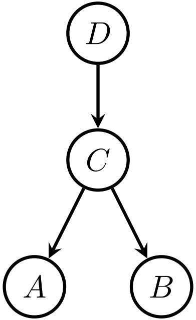


...compile a leaf CPT



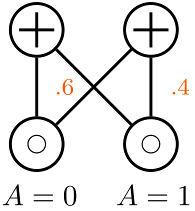
From tree BN to circuits

via compilation



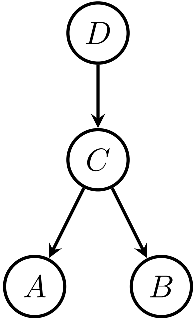
...compile a leaf CPT

$p(A|C=1)$

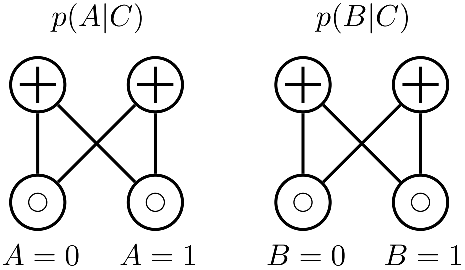


From tree BN to circuits

via compilation

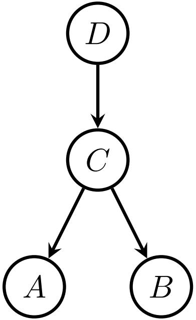


...compile a leaf CPT...for all leaves...

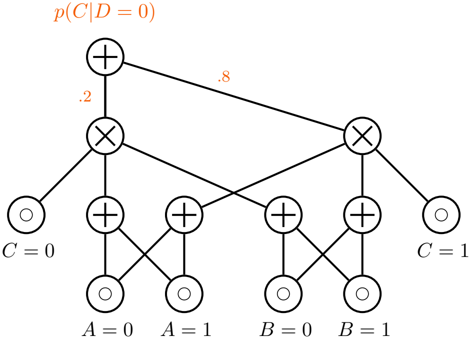


From tree BN to circuits

via compilation

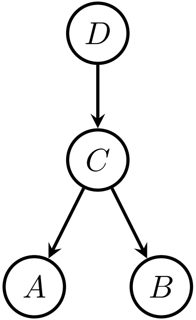


...and recurse over parents...

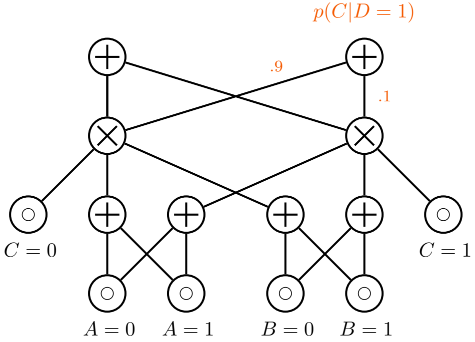


From tree BN to circuits

via compilation

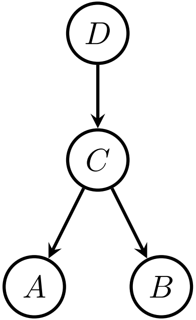


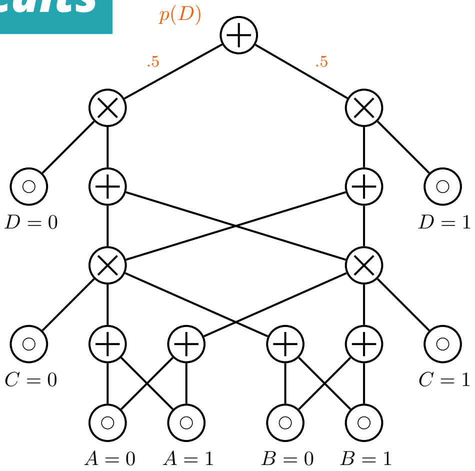
...while reusing previously compiled nodes!...



From tree BN to circuits

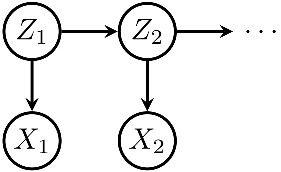
via compilation

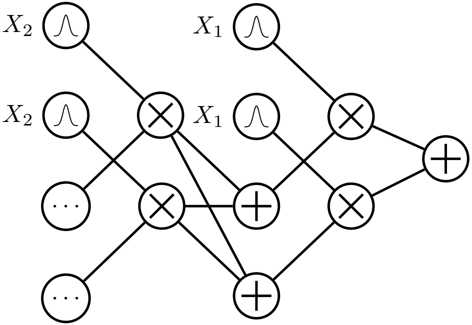




Hidden Markov Models

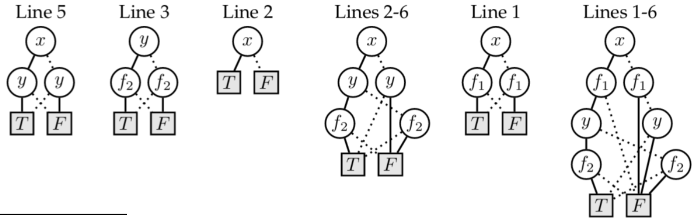
as computational graphs





Compilation : probabilistic programming

```
1  x = flip(θ1);
2  if(x) {
3    y = flip(θ2)
4  } else {
5    y = x
6  }
```



Chavira et al., "Compiling relational Bayesian networks for exact inference", 2006

Holtzen et al., "Symbolic Exact Inference for Discrete Probabilistic Programs", 2019

De Raedt et al.; Riguzzi; Fierens et al.; Vlasselaer et al., "ProbLog: A Probabilistic Prolog and Its Application in Link Discovery."; "A top down interpreter for LPAD and CP-logic"; "Inference and Learning in Probabilistic Logic Programs using Weighted Boolean Formulas"; "Anytime Inference in Probabilistic Logic Programs with Tp-compilation", 2007; 2007; 2015; 2015

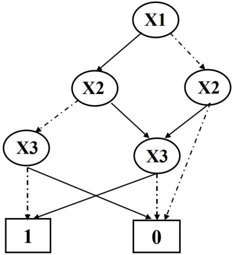
Olteanu et al.; Van den Broeck et al., "Using OBDDs for efficient query evaluation on probabilistic databases"; Query Processing on Probabilistic Data: A Survey, 2008; 2017

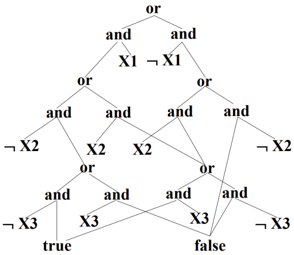
Vlasselaer et al., "Exploiting Local and Repeated Structure in Dynamic Bayesian Networks", 2016

Decision Diagrams

- FBDDs (Free binary decision diagrams; *read-once*)
- OBDDs (Ordered BDDs)
- SDDs (Sentential decision diagrams)

$\Longrightarrow$ *BDD as circuit*

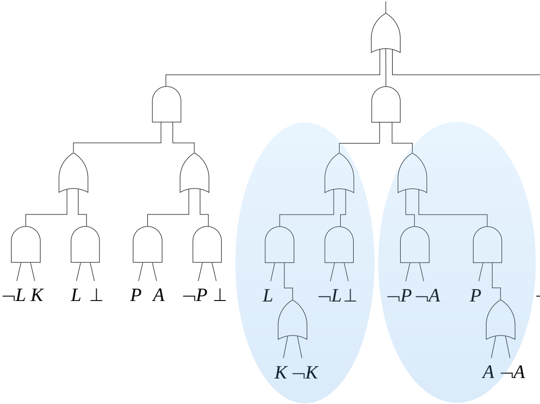


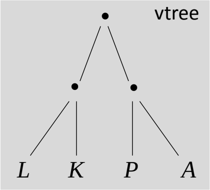


Darwiche and Marquis, “A knowledge compilation map”, 2002

Structured Decomposability

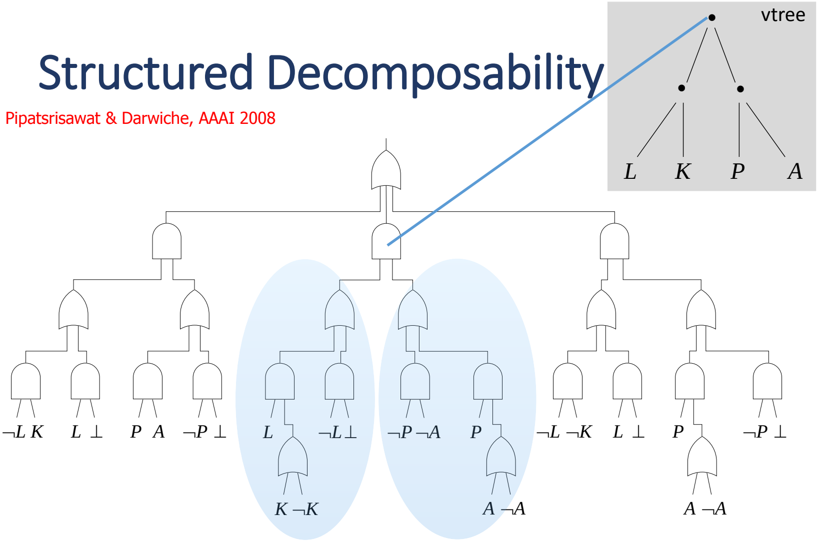
Pipatsrisawat & Darwiche, AAAI 2008





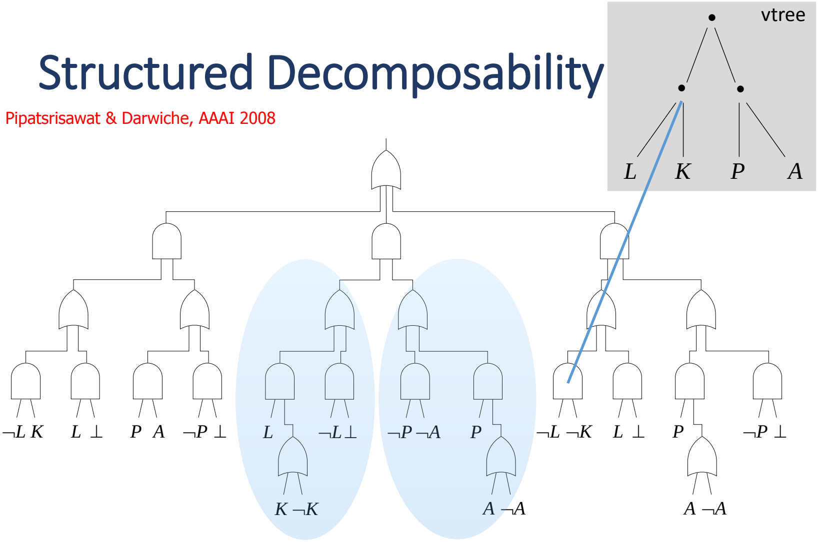
Structured Decomposability

Pipatsrisawat & Darwiche, AAAI 2008



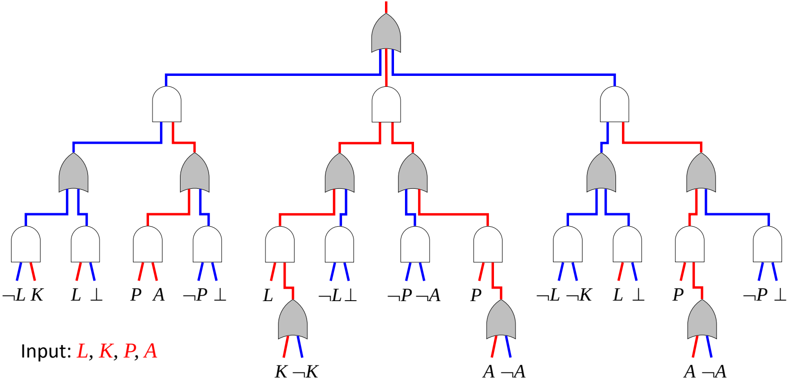
Structured Decomposability

Pipatsrisawat & Darwiche, AAAI 2008



Partitioned Determinism (SDDs)

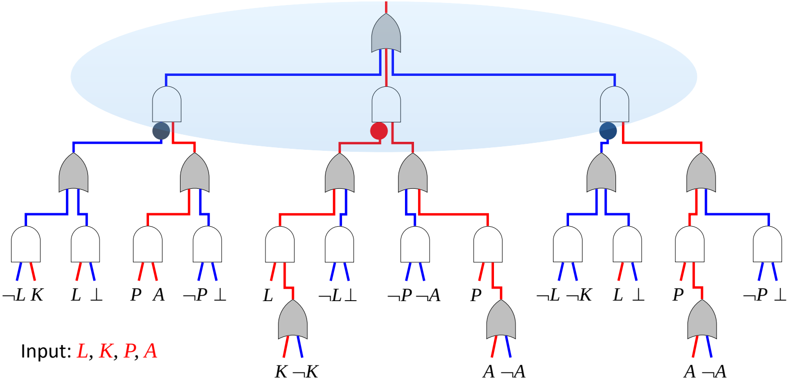
Darwiche, IJCAI 2011



Input: L, K, P, A

Partitioned Determinism (SDDs)

Darwiche, IJCAI 2011

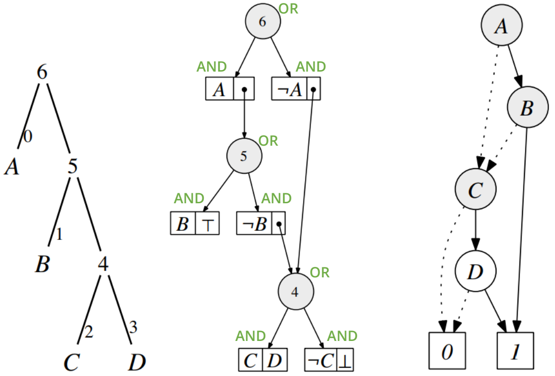


Input: L, K, P, A

Decision Diagrams

- FBDDs (Free binary decision diagrams; *read-once*)
- OBDDs (Ordered BDDs)
- SDDs (Sentential decision diagrams)

$\Longrightarrow$ *SDD & OBDD for*
$(A \wedge B) \vee (C \wedge D)$

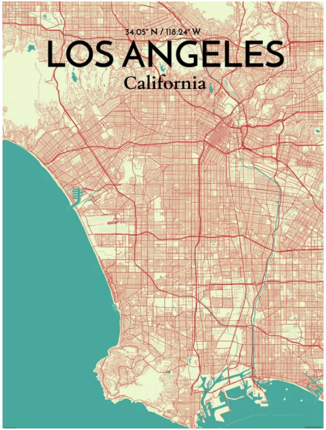


(a) vtree (b) SDD (c) OBDD

Darwiche, "SDD: A new canonical representation of propositional knowledge bases", 2011

Probability of logical events

q_8: *What is the probability of having a traffic jam on my route to campus?*



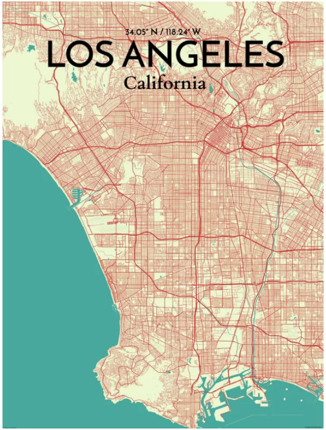
© fineartamerica.com

Probability of logical events

q_8: *What is the probability of having a traffic jam on my route to campus?*

$$q_8(\mathbf{m}) = p_{\mathbf{m}}\left(\bigvee_{i \in \mathsf{route}} \mathsf{Jam}_{\mathsf{Str}\ i}\right)$$

$\Longrightarrow$ ***marginals + logical events***

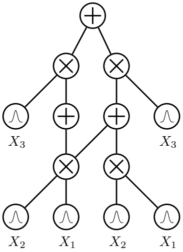


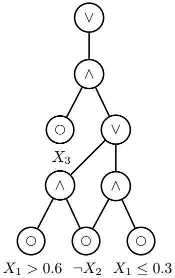
Smoothness + structured decomp. = tractable PR

Computing $p(\alpha)$: the probability of arbitrary logical formula

Multilinear in circuit sizes if the logical circuit:

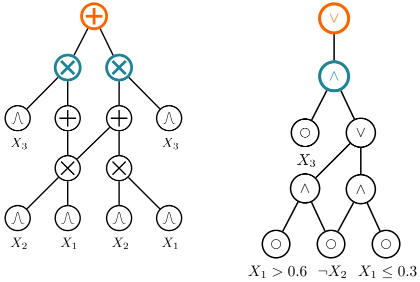
- is smooth, structured decomposable, deterministic
- shares the **same vtree**





Smoothness + structured decomp. = tractable PR

If $p(\mathbf{x}) = \sum_i w_i p_i(\mathbf{x})$, $\alpha = \bigvee_j \alpha_j$,
(***smooth*** p)
(***smooth + deterministic*** α):



$$p(\alpha) = \sum_i w_i p_i \left(\bigvee_j \alpha_j \right) = \sum_i w_i \sum_j p_i \left(\alpha_j \right)$$

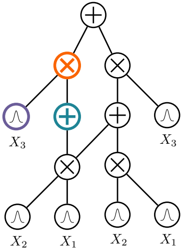
$\Rightarrow$ *probabilities are "pushed down" to children*

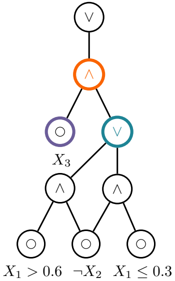
Smoothness + **structured decomp.** = **tractable PR**

If $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y})$, $\alpha = \beta \wedge \gamma$,
(***structured decomposability***):

$$p(\alpha) = p(\beta \wedge \gamma) \cdot p(\beta \wedge \gamma) = p(\beta) \cdot p(\gamma)$$

$\Longrightarrow$ *probabilities decompose into simpler ones*





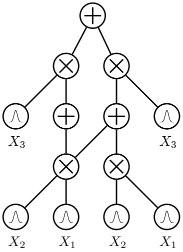
Smoothness + **structured decomp.** = **tractable PR**

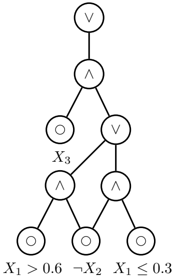
To compute $p(\alpha)$:

- compute the probability for each **pair** of probabilistic and logical circuit nodes for the **same vtree node**

$\Longrightarrow$ *cache the values!*

- feedforward evaluation (bottom-up)





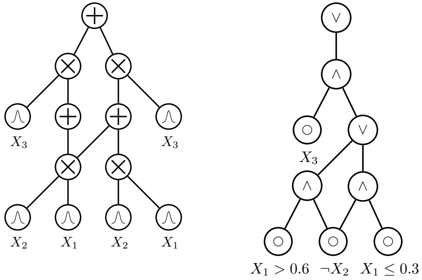
Smoothness + **structured decomp.** = **tractable PR**

To compute $p(\alpha)$:

- compute the probability for each **pair** of probabilistic and logical circuit nodes for the **same vtree node**

  $\Longrightarrow$ *cache the values!*

- feedforward evaluation (bottom-up)



structured decomposability = *tractable...*

- ***Symmetric*** and ***group queries*** (exactly-k, odd-number, etc.) *(Bekker et al. 2015)*

For the “right” vtree

- Marginal MAP *(Oztok et al. 2016)*
- Probability of logical circuit event in probabilistic circuit *(Choi et al. 2015b)*
- ***Multiply*** two probabilistic circuits *(Shen et al. 2016)*
- ***KL Divergence*** between probabilistic circuits *(Liang et al. 2017)*
- ***Same-decision probability*** *(Oztok et al. 2016)*
- ***Expected same-decision probability*** *(Choi et al. 2017)*
- ***Expected classifier agreement*** *(Choi et al. 2018)*
- ***Expected predictions*** *(Khosravi et al. 2019b)*

Questions answered today

1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. ***How are circuit size and tractability related?*** (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. *Are all tractable distributions probabilistic circuits?* (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

Questions answered today

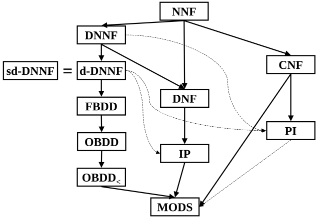
1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. ***How are circuit size and tractability related?*** (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. *Are all tractable distributions probabilistic circuits?* (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

Succinctness of circuits

Expressive efficiency

Tractability is defined with respect to the size of the model.

How do structural constraints affect **expressive efficiency** (**succinctness**) of probabilistic/logical circuits?



Succinctness of circuits

Expressive efficiency

A family of circuits $\mathcal{M}_1$ *is* ***at least as succinct as*** $\mathcal{M}_2$
iff for every $\mathbf{m}_2 \in \mathcal{M}_2$*, there exists* $\mathbf{m}_1 \in \mathcal{M}_1$ *that represents the same function and* $|m_1| \leq |\mathsf{poly}(m_2)|$.

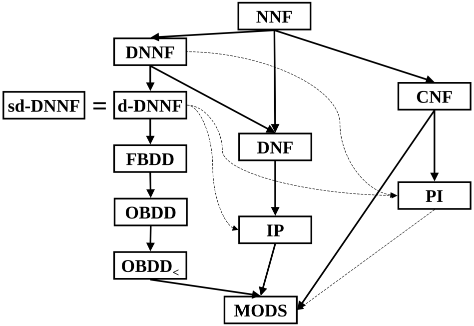
$\Rightarrow$ *denoted* $\mathcal{M}_1 \leq \mathcal{M}_2$

$\Rightarrow$ *strictly more succinct* $(\mathcal{M}_1 < \mathcal{M}_2)$
iff $\mathcal{M}_1 \leq \mathcal{M}_2$ *and* $\mathcal{M}_1 \not\geq \mathcal{M}_2$

Succinctness of circuits

Expressive efficiency

Strict succinctness ordering: DNNF < d-DNNF < FBDD < OBDD



Darwiche and Marquis, "A knowledge compilation map", 2002

Succinctness of circuits

Expressive efficiency

Strict succinctness ordering: **DNNF < d-DNNF** < FBDD < OBDD

- d-DNNF $\not\leq$ DNNF unless the polynomial hierarchy collapses *(Darwiche et al. 2002a)*.
- The *Sauerhoff function* has DNNF of size $O(n^2)$ but d-DNNF of size $2^{\Omega(n)}$ *(Bova et al. 2016)*.

Succinctness of circuits

Expressive efficiency

Strict succinctness ordering: **DNNF < d-DNNF** < FBDD < OBDD

- d-DNNF $\not\leq$ DNNF unless the polynomial hierarchy collapses *(Darwiche et al. 2002a).*
- The *Sauerhoff function* has DNNF of size $O(n^2)$ but d-DNNF of size $2^{\Omega(n)}$ *(Bova et al. 2016).*

$\Longrightarrow$ *Unconditional exponential separation for d-DNNF $\not\leq$ DNNF*

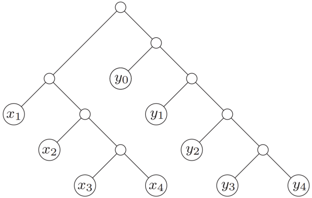
$\Longrightarrow$ *Using a connection between circuits and* ***communication complexity***

Succinctness of circuits

Expressive efficiency

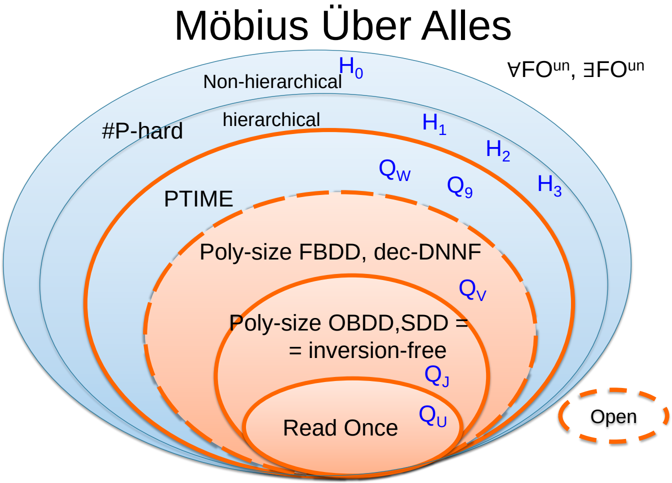
SDD < OBDD: *SDDs are strictly more succinct than OBDDs*

- SDD $\leq$ OBDD: OBDDs are SDDs with right-linear vtrees
- SDD $\not\geq$ OBDD: The *hidden weighted bit function* has SDD of size $O(n^3)$ but OBDD of size $2^\Omega(n)$.



Bova, "SDDs are exponentially more succinct than OBDDs", 2016

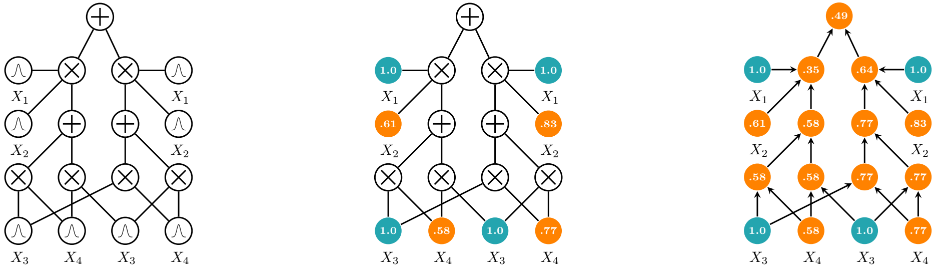
Query compilation



How precise is the characterization of tractable circuits by structural properties?

Smoothness + decomposability = tractable MAR

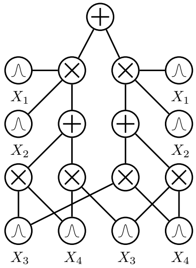
Recall: Smoothness and decomposability allow marginal inference by feedforward (sum-product) evaluation.

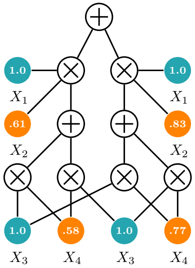


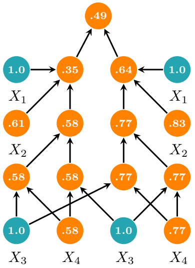
Smoothness + **decomposability** = **tractable MAR**

Recall: Smoothness and decomposability allow marginal inference by feedforward (sum-product) evaluation.

$\Rightarrow$ *Are these properties necessary?*





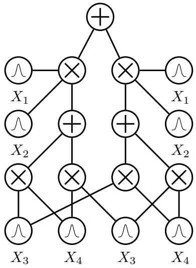


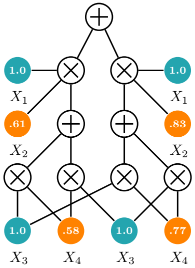
Smoothness + decomposability = tractable MAR

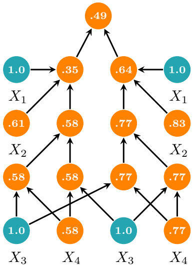
Recall: Smoothness and decomposability allow marginal inference by feedforward (sum-product) evaluation.

$\Rightarrow$ *Are these properties necessary?*

$\Rightarrow$ *Yes! Otherwise, integrals do not decompose.*

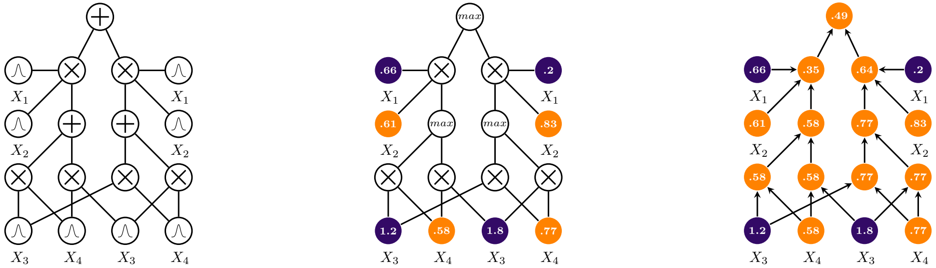






Determinism + **decomposability** = **tractable MAP**

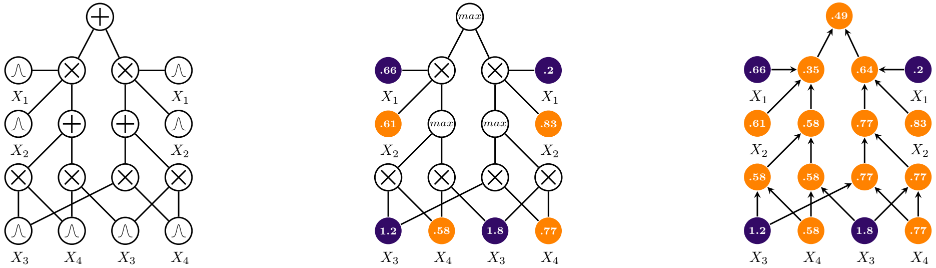
Recall: Determinism and decomposability allow MAP inference by feedforward (max-product) evaluation.



Determinism + decomposability = tractable MAP

Recall: Determinism and decomposability allow MAP inference by feedforward (max-product) evaluation.

$\Rightarrow$ *However, decomposability is not necessary!*

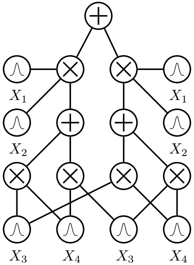


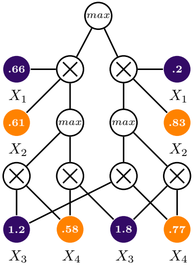
Determinism + decomposability = tractable MAP

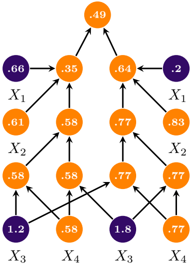
Recall: Determinism and decomposability allow MAP inference by feedforward (max-product) evaluation.

⇒ *However, decomposability is not necessary!*

⇒ *A weaker condition, **consistency**, suffices.*



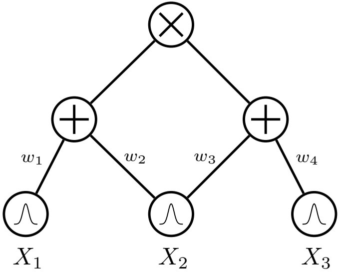




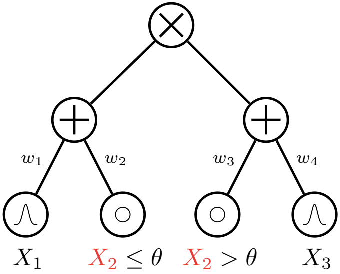
Consistency

A product node is consistent if any variable shared between its children appears in a single leaf node

$\Rightarrow$ *decomposability implies consistency*



consistent circuit



inconsistent circuit

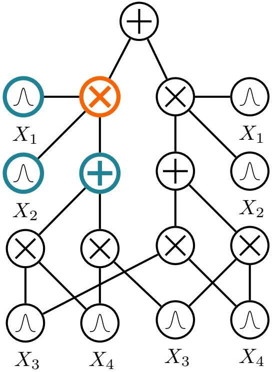
Determinism + **consistency** = **tractable MAP**

Determinism + **consistency** = **tractable MAP**

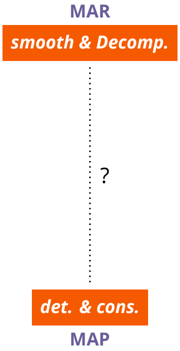
If $\max_{\mathbf{q}_{\text{shared}}} p(\mathbf{q}, \mathbf{e}) = \max_{\mathbf{q}_{\text{shared}}} p(\mathbf{q}_{\mathbf{x}}, \mathbf{e}_{\mathbf{x}}) \cdot \max_{\mathbf{q}_{\text{shared}}} p(\mathbf{q}_{\mathbf{y}}, \mathbf{e}_{\mathbf{y}})$ (***consistent***):

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

$\Longrightarrow$ *solving optimization independently*

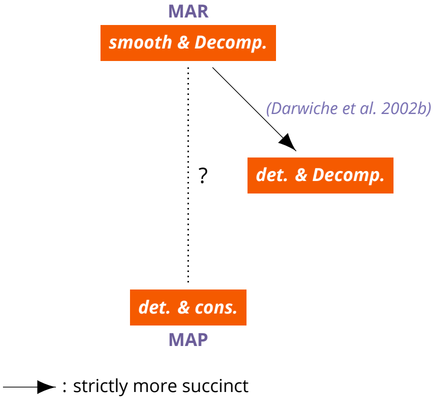


Expressive efficiency of circuits



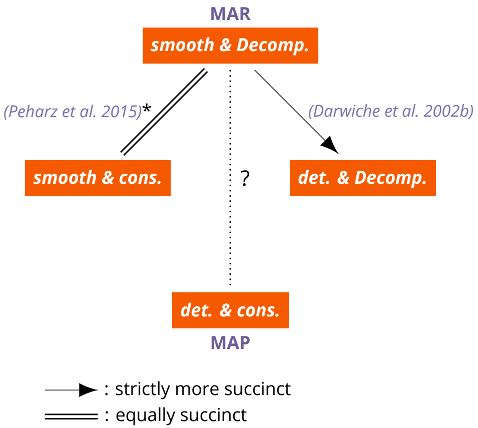
Are smooth & decomposable circuits as succinct as deterministic & consistent ones, or vice versa?

Expressive efficiency of circuits



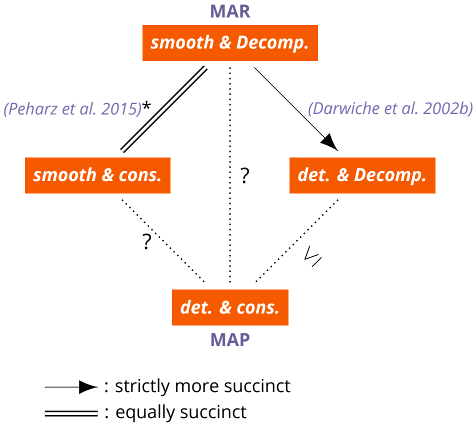
- Smooth & decomposable circuits strictly more succinct than deterministic & decomposable ones
- Smooth & consistent circuits are equally succinct as smooth & decomposable ones

Expressive efficiency of circuits



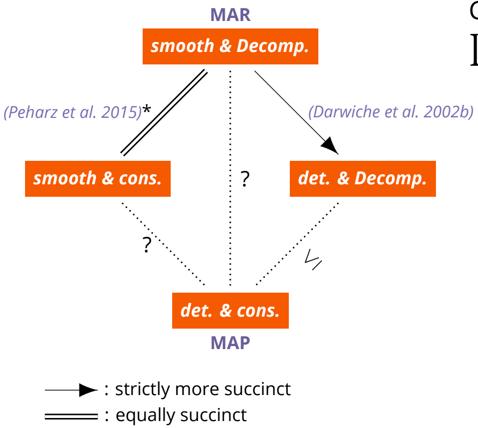
- Smooth & decomposable circuits strictly more succinct than deterministic & decomposable ones
- Smooth & consistent circuits are equally succinct as smooth & decomposable ones

Expressive efficiency of circuits



- Smooth & decomposable circuits strictly more succinct than deterministic & decomposable ones
- Smooth & consistent circuits are equally succinct as smooth & decomposable ones

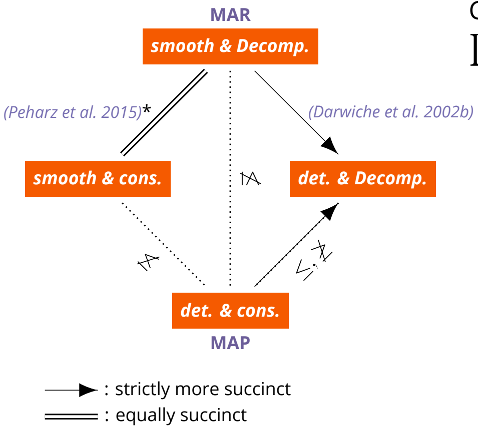
Expressive efficiency of circuits



Consider following circuit over Boolean variables:
$\prod_i^r (Y_i \cdot Z_{i1} + (\neg Y_i) \cdot Z_{i2}), \quad Z_{ij} \in \mathbf{X}$

- Size linear in the number of variables
- Deterministic and consistent
- Marginal (with no evidence) is the solution to #P-hard SAT′ problem *(Valiant 1979)* $\Rightarrow$ **no tractable circuit for marginals!**

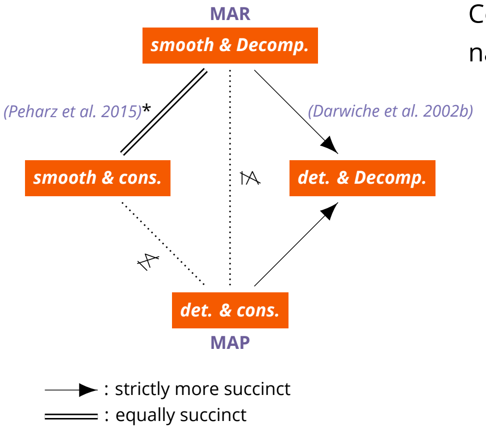
Expressive efficiency of circuits



Consider following circuit over Boolean variables:
$\prod_i^r (Y_i \cdot Z_{i1} + (\neg Y_i) \cdot Z_{i2}), \quad Z_{ij} \in \mathbf{X}$

- Size linear in the number of variables
- Deterministic and consistent
- Marginal (with no evidence) is the solution to #P-hard SAT$'$ problem *(Valiant 1979)* $\Rightarrow$ **no tractable circuit for marginals!**

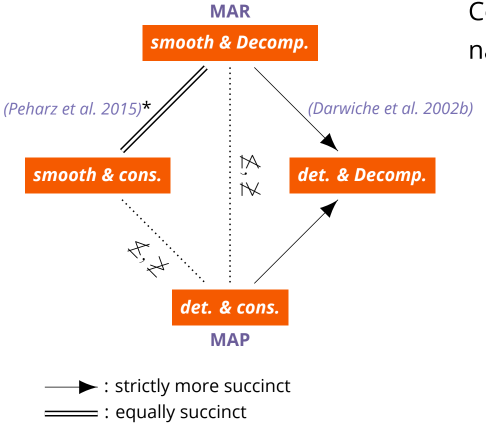
Expressive efficiency of circuits



Consider the marginal distribution $p(\mathbf{X})$ from a naive Bayes distribution $p(\mathbf{X}, C)$:

- Linear-size smooth and decomposable circuit
- MAP of $p(\mathbf{X})$ solves marginal MAP of $p(\mathbf{X}, C)$ which is NP-hard *(de Campos 2011)*
  $\Rightarrow$ **no tractable circuit for MAP!**

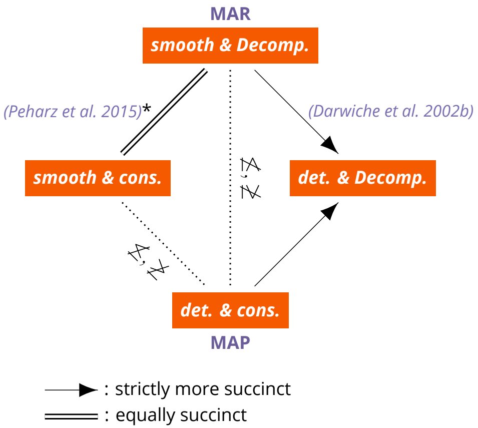
Expressive efficiency of circuits



Consider the marginal distribution $p(\mathbf{X})$ from a naive Bayes distribution $p(\mathbf{X}, C)$:

- Linear-size smooth and decomposable circuit
- MAP of $p(\mathbf{X})$ solves marginal MAP of $p(\mathbf{X}, C)$ which is NP-hard *(de Campos 2011)*
  $\Rightarrow$ **no tractable circuit for MAP!**

Expressive efficiency of circuits



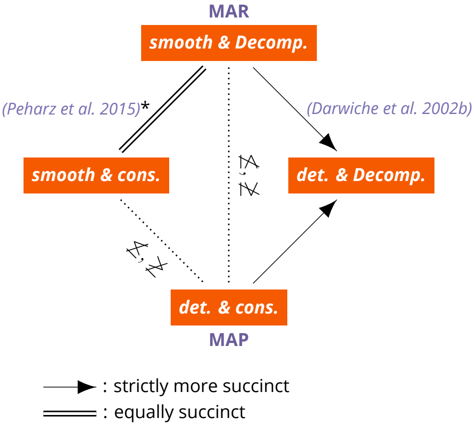
- Neither smooth & decomposable nor deterministic & consistent circuits are more succinct than the other!

  $\Rightarrow$ *Choose tractable circuit family based on your query*

- More theoretical questions remaining

  $\Rightarrow$ *"Complete the map"*

Expressive efficiency of circuits



- Neither smooth & decomposable nor deterministic & consistent circuits are more succinct than the other!

  $\Longrightarrow$ *Choose tractable circuit family based on your query*

- More theoretical questions remaining

  $\Longrightarrow$ *"Complete the map"*

Expressive efficiency of circuits

$$
\begin{tikzcd}
 & \text{sdwD-AC}_m \arrow[rr, equal] & & \text{sdD-AC}_m \arrow[dd, equal] \\
\text{swD-AC}_m \arrow[ur] \arrow[rr, equal] & & \text{sD-AC}_m \arrow[ur] \arrow[dd, equal] & \\
 & \text{dwD-AC}_m \arrow[uu] \arrow[rr] & & \text{dD-AC}_m \\
\text{wD-AC}_m \arrow[uu] \arrow[ur] \arrow[rr] & & \text{D-AC}_m \arrow[ur] &
\end{tikzcd}
$$

Succinctness map for *monotone* circuits

$\Longrightarrow$ *(s)mooth, (d)eterministic, (D)ecomposable, (w)eak (D)ecomposable (i.e. consistent)*

Expressive efficiency of circuits

$$\begin{tikzcd}
 & \mathrm{sdwD\text{-}AC}_m \arrow[rr, equal] & & \mathrm{sdD\text{-}AC}_m \arrow[dd, equal] \\
\mathrm{swD\text{-}AC}_m \arrow[ur] \arrow[rr, equal] & & \mathrm{sD\text{-}AC}_m \arrow[ur] \arrow[dd, equal] & \\
 & \mathrm{dwD\text{-}AC}_m \arrow[uu] \arrow[rr] & & \mathrm{dD\text{-}AC}_m \\
\mathrm{wD\text{-}AC}_m \arrow[uu] \arrow[ur] \arrow[rr] & & \mathrm{D\text{-}AC}_m \arrow[ur] &
\end{tikzcd}$$

Succinctness map for *monotone* circuits

$$\begin{tikzcd}
 & \mathrm{sdwD\text{-}AC}_p \arrow[rr, equal] & & \mathrm{sdD\text{-}AC}_p \arrow[dd, equal] \\
\mathrm{swD\text{-}AC}_p \arrow[ur] \arrow[rr, equal] & & \mathrm{sD\text{-}AC}_p \arrow[ur] \arrow[dd, equal] & \\
 & \mathrm{dwD\text{-}AC}_p \arrow[uu] \arrow[rr] \arrow[dr, dotted, no head, "?"] & & \mathrm{dD\text{-}AC}_p \\
\mathrm{wD\text{-}AC}_p \arrow[uu, dotted, no head, "?"] \arrow[ur] \arrow[rr, dotted, no head, "?"] & & \mathrm{D\text{-}AC}_p \arrow[ur] &
\end{tikzcd}$$

Succinctness map for *positive* circuits (non-negative output, but weights may be negative)

$\Longrightarrow$ *(s)mooth, (d)eterministic, (D)ecomposable, (w)eak (D)ecomposable (i.e. consistent)*

Colnet and Mengel, "A Compilation of Succinctness Results for Arithmetic Circuits", 2021

Questions answered today

1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. ***What's the most impressive query we can efficiently compute?*** (YooJung)
7. *Are all tractable distributions probabilistic circuits?* (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

Questions answered today

1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. ***What's the most impressive query we can efficiently compute?*** (YooJung)
7. *Are all tractable distributions probabilistic circuits?* (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

Goal

Given a class of queries can we systematically find a class of probabilistic circuits that is tractable for it?

A language for queries

Integral expressions that can be formed by composing these operators

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

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

A language for queries

Integral expressions that can be formed by composing these operators

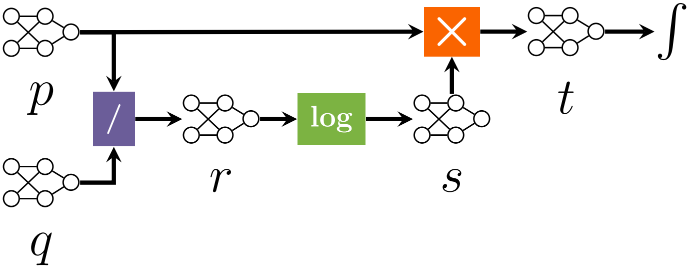
$+$, $\times$, **pow**, **log**, **exp** and $/$

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

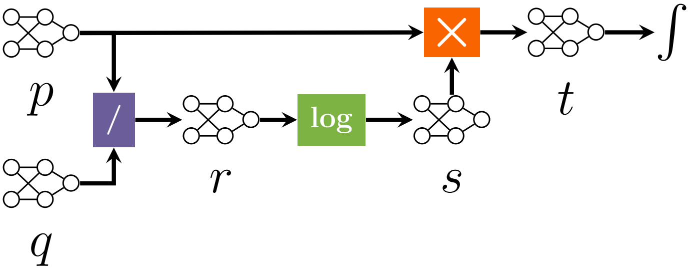
Represented as ***higher-order computational graphs***—pipelines—operating over circuits!

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

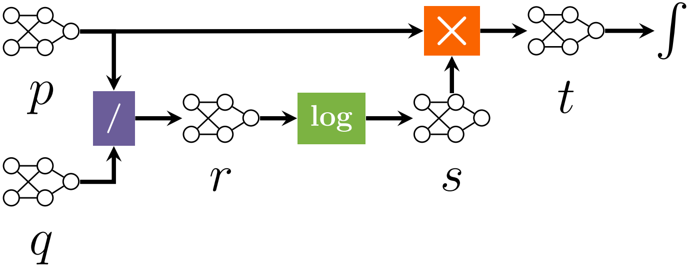
$$\mathbb{KLD}(p \mid\mid q) = \int_{\mathsf{val}(\mathbf{X})} p(\mathbf{x}) \times \log\left(p(\mathbf{x})/q(\mathbf{x})\right) \; d\mathbf{X}$$



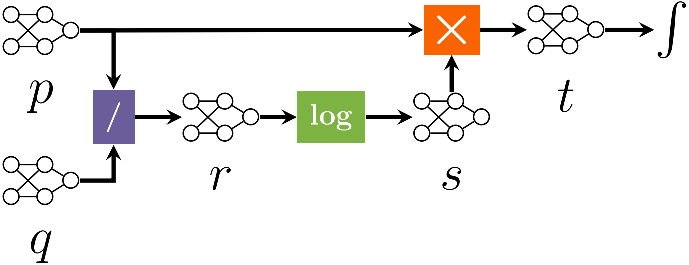
$$\mathbb{KLD}(p \mid\mid q) = \int_{\mathsf{val}(\mathbf{X})} p(\mathbf{x}) \times \log\left(p(\mathbf{x}) \boxed{/}\, q(\mathbf{x})\right)\, d\mathbf{X}$$



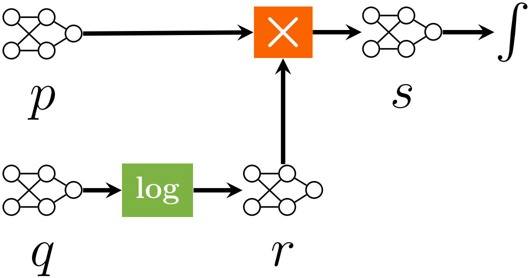
$$\mathbb{KLD}(p \mid\mid q) = \int_{\mathsf{val}(\mathbf{X})} p(\mathbf{x}) \times \boxed{\mathbf{log}} \, (p(\mathbf{x})/q(\mathbf{x})) \;\, d\mathbf{X}$$



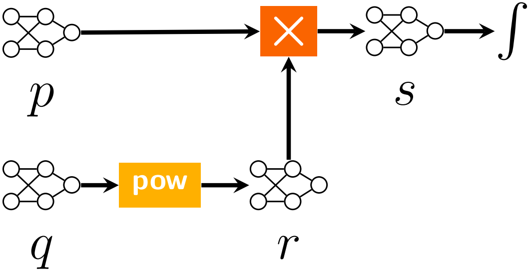
$$\mathbb{KLD}(p \,||\, q) = \int_{\mathsf{val}(\mathbf{X})} p(\mathbf{x}) \times \log\left(p(\mathbf{x})/q(\mathbf{x})\right) \; d\mathbf{X}$$



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



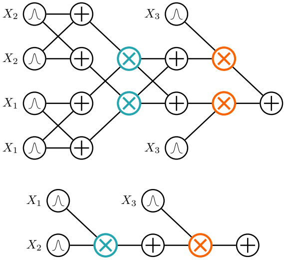
$$\mathbb{E}_{\mathbf{x}^m \sim p(\mathbf{x}^m | \mathbf{x}^o)} \left[q^\alpha(\mathbf{x}^m, \mathbf{x}^o) \right]$$



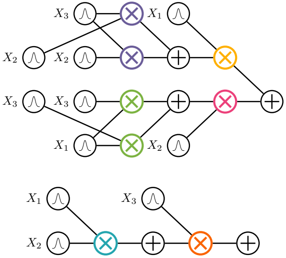
Compatibility

Two circuits are *compatible* if they have the same *hierarchical scope partitioning*

$\Longrightarrow$ *generalizes "structured decomposability with same vtree"*

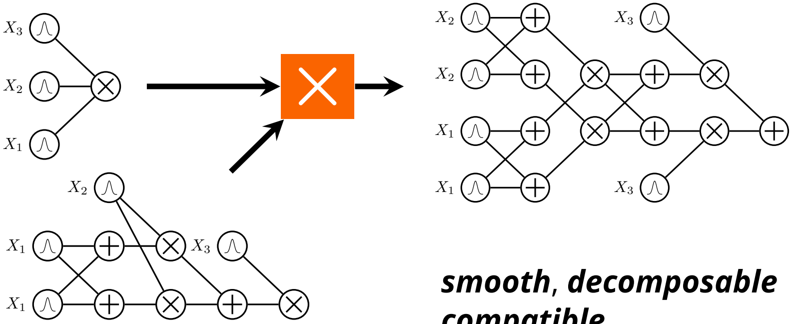


compatible circuits



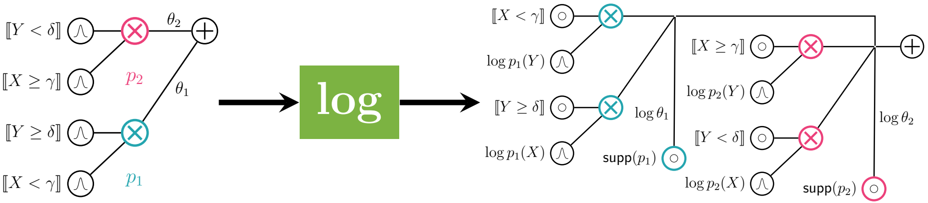
non-compatible circuits

Tractable operators



smooth, decomposable compatible

Tractable operators

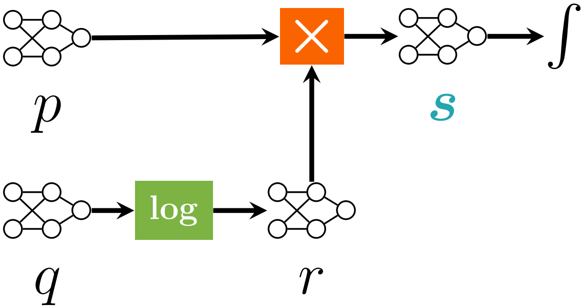


smooth, decomposable deterministic

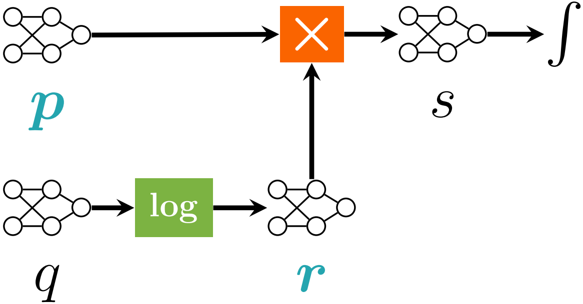
smooth, decomposable

		× $p \times q$	+ $p + q$	$\mathrm{pow}_{\mathbb{N}}$ p^n	$\mathrm{pow}_{\mathbb{R}}$ p^α	/ p/q	log $\log p$	exp $\exp p$
smoothness	**SMO**	✗	✓	✓	✓	✓	✓	✓
decomposability	**DEC**	✗	✗	✗	✓	✗	✗	✗
determinism	**DET**	✗	✗	✗	✓	✓	✓	✗
compatibility	**CMP**	✗	✓	✓	✗	✓	✗	✗

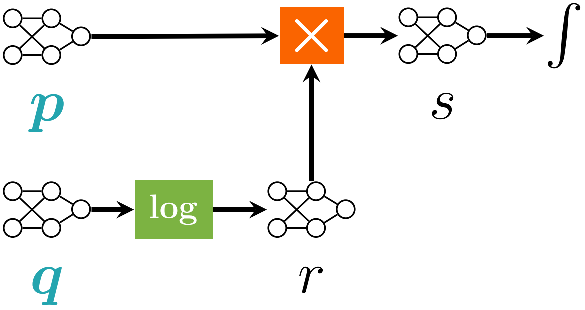
Building an atlas of composable ***tractable atomic operations***



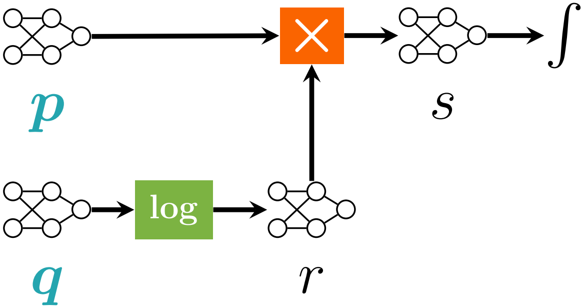
To perform tractable integration we need $\boldsymbol{s}$ to be ***smooth and decomposable...***



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



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



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

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

***compositionally** derive the tractability of many more queries*

***Vergari** et al., "A Compositional Atlas of Tractable Circuit Operations: From Simple Transformations to Complex Information-Theoretic Queries", 2021*

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

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

Vergari *et al., "A Compositional Atlas of Tractable Circuit Operations: From Simple Transformations to Complex Information-Theoretic Queries", 2021*

Composable tractable sub-routines

```
function kld(p, q)
    r = quotient(p, q)
    s = log(r)
    t = product(p, s)
    return integrate(t)
end
```

```
function xent(p, q)
    r = log(q)
    s = product(p, r)
    return -integrate(s)
end
```

```
function ent(p)
    q = log(p)
    r = product(p, q)
    return -integrate(s)
end
```

```
function alphadiv(p, q, alpha=1.5)
    r = real_pow(p, alpha)
    s = real_pow(q, 1.0-alpha)
    t = product(r, s)
    return log(integrate(t)) / (1.0-alpha)
end
```

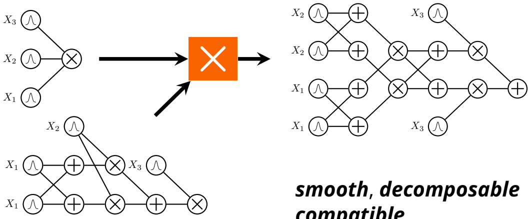
*Efficient inference algorithms **in a couple lines of Julia code!*** [2]

[2] https://github.com/UCLA-StarAI/circuit-ops-atlas

Next up...

1. Learning and reasoning with symbolic constraints
2. Expected predictions: handling missing values, fairness
3. Exact inference of causal effects

$\Rightarrow$ *using tractable operators*



smooth**, **decomposable
compatible

Symbolic constraints

"How can neural nets reason and learn with symbolic constraints reliably and efficiently?"

When?

Ground Truth

e.g. predict shortest path in a map

When?

Ground Truth

given $\mathbf{x}$ *// e.g. a tile map*

structured output prediction (SOP) tasks

Vlastelica et al., "Differentiation of blackbox combinatorial solvers", 2020

When?

Ground Truth

given $\mathbf{x}$ *// e.g. a tile map*

find $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}} p_\theta(\mathbf{y} \mid \mathbf{x})$ *// e.g. a configurations of edges in a grid*

structured output prediction (SOP) tasks

Vlastelica et al., "Differentiation of blackbox combinatorial solvers", 2020

When?

Ground Truth

given $\mathbf{x}$ *// e.g. a tile map*

find $\mathbf{y}^* = \mathrm{argmax}_{\mathbf{y}}\, p_\theta(\mathbf{y} \mid \mathbf{x})$ *// e.g. a configurations of edges in a grid*

s.t. $\mathbf{y} \models \mathsf{K}$ *// e.g., that form a valid path*

structured output prediction (SOP) tasks

Vlastelica et al., "Differentiation of blackbox combinatorial solvers", 2020

When?

Ground Truth

given $\mathbf{x}$ *// e.g. a tile map*

find $\mathbf{y}^* = \text{argmax}_{\mathbf{y}}\, p_\theta(\mathbf{y} \mid \mathbf{x})$ *// e.g. a configurations of edges in a grid*

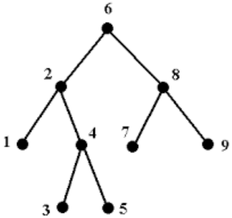
s.t. $\mathbf{y} \models \mathsf{K}$ *// e.g., that form a valid path*

// for a 12×12 grid, 2^{144} states but only 10^{10} valid ones!

structured output prediction (SOP) tasks

Vlastelica et al., "Differentiation of blackbox combinatorial solvers", 2020

When?



given $\mathbf{x}$ *// e.g. a feature map*

find $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}} p_\theta(\mathbf{y} \mid \mathbf{x})$ *// e.g. labels of classes*

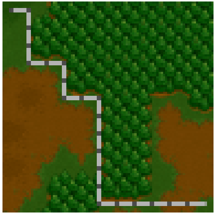
s.t. $\mathbf{y} \models \mathsf{K}$ *// e.g., constraints over superclasses*

$$\mathsf{K} : (Y_{\mathsf{cat}} \implies Y_{\mathsf{animal}}) \land (Y_{\mathsf{dog}} \implies Y_{\mathsf{animal}})$$

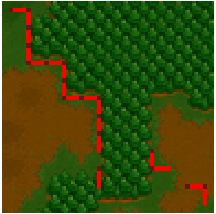
hierarchical multi-label classification

Giunchiglia and Lukasiewicz, "Coherent hierarchical multi-label classification networks", 2020

When?



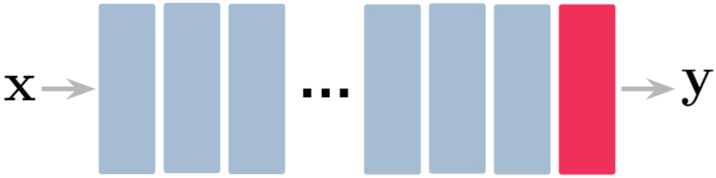
Ground Truth



ResNet-18

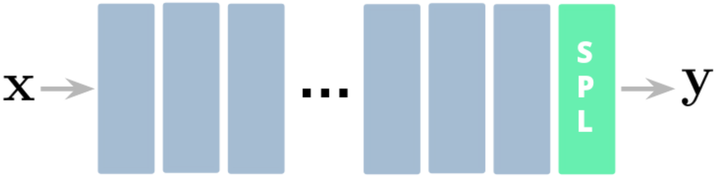
neural nets struggle to satisfy domain constraints!

How?



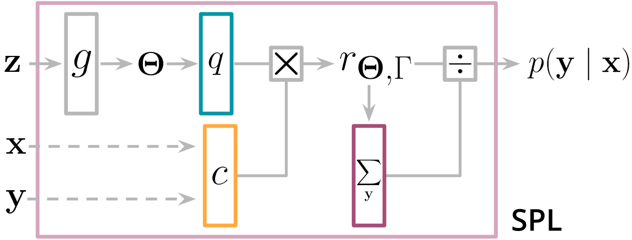
take an unreliable neural network architecture...

How?



......and replace the last layer with a semantic probabilistic layer

SPL

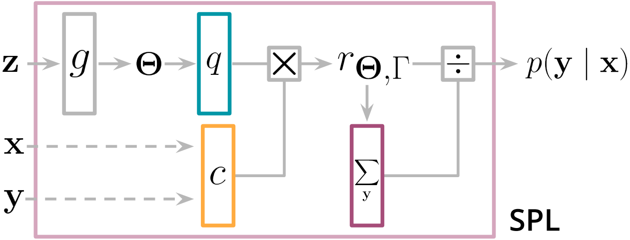


$q_{\Theta}(\mathbf{y} \mid g(\mathbf{z}))$ is an expressive distribution over labels

$c_{\mathsf{K}}(\mathbf{x}, \mathbf{y})$ encodes the constraint $\mathbb{1}\{\mathbf{x}, \mathbf{y} \models \mathsf{K}\}$

Ahmed et al., "Semantic Probabilistic Layers for Neuro-Symbolic Learning", 2022

SPL

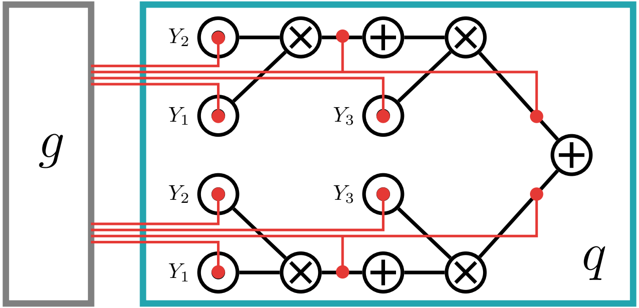


$$p(\mathbf{y} \mid \mathbf{x}) = q_{\Theta}(\mathbf{y} \mid g(\mathbf{z})) \cdot c_{\mathsf{K}}(\mathbf{x}, \mathbf{y}) / \mathcal{Z}(\mathbf{x})$$

$$\mathcal{Z}(\mathbf{x}) = \sum_{\mathbf{y}} q_{\Theta}(\mathbf{y} \mid \mathbf{x}) \cdot c_{\mathsf{K}}(\mathbf{x}, \mathbf{y})$$

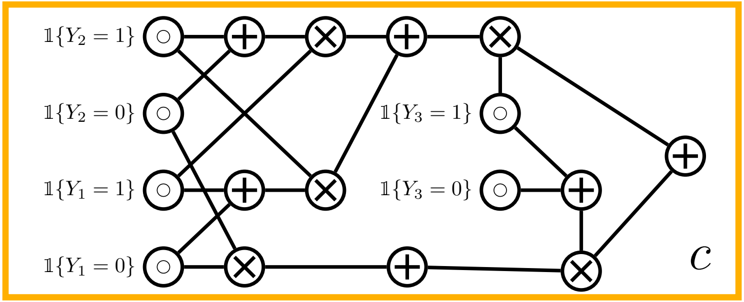
Ahmed et al., "Semantic Probabilistic Layers for Neuro-Symbolic Learning", 2022

SPL



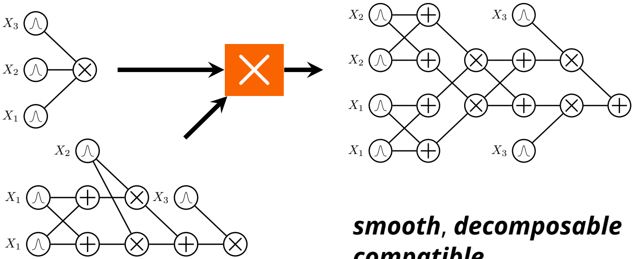
a conditional circuit $q(\mathrm{y}; \Theta = g(\mathrm{z}))$

SPL



and a logical circuit $c(\mathrm{y}, \mathrm{x})$ **encoding** K

Tractable products



smooth, decomposable compatible

exactly compute $\mathcal{Z}$ in time $O(|\boldsymbol{q}||\boldsymbol{c}|)$

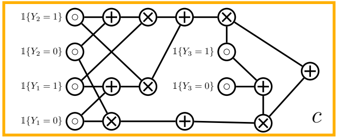
SPL recipe

$$
\begin{aligned}
\mathsf{K}: &\ (Y_1 = 1 \implies Y_3 = 1) \\
\wedge &\ (Y_2 = 1 \implies Y_3 = 1)
\end{aligned}
$$

1) Take any
logical constraint

SPL recipe

$$\mathsf{K}: (Y_1 = 1 \implies Y_3 = 1)$$
$$\wedge \quad (Y_2 = 1 \implies Y_3 = 1)$$

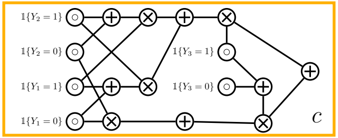


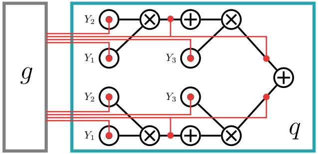
1) Take any logical constraint

2) Compile it into a constraint circuit

SPL recipe

$$K: (Y_1 = 1 \implies Y_3 = 1)$$
$$\wedge \quad (Y_2 = 1 \implies Y_3 = 1)$$





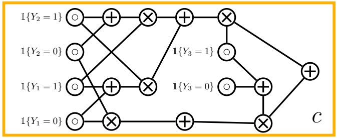
1) Take any logical constraint

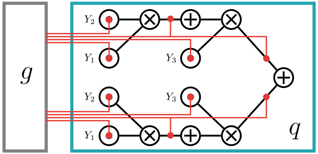
2) Compile it into a constraint circuit

3) Multiply it by a circuit distribution

SPL recipe

$$K: (Y_1 = 1 \implies Y_3 = 1)$$
$$\land \quad (Y_2 = 1 \implies Y_3 = 1)$$





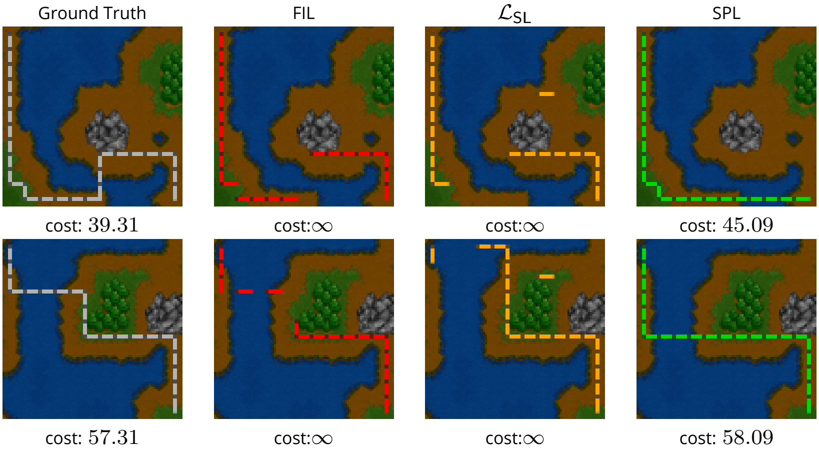
1) Take any logical constraint

2) Compile it into a constraint circuit

3) Multiply it by a circuit distribution

4) train end-to-end by sgd!

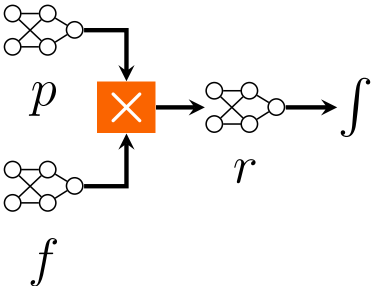
Guaranteeing consistency



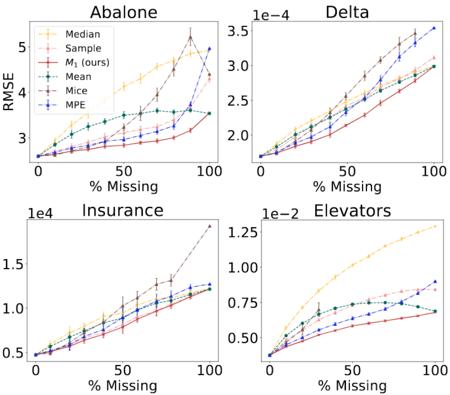
Expected predictions

Reasoning about the output of a classifier or regressor $\boldsymbol{f}$ given a distribution $\boldsymbol{p}$ over the input features

$$\mathbb{E}_p[f] = \int_{\mathsf{val}(\mathbf{X})} p(\mathbf{x}) \times f(\mathbf{x})\, d\mathbf{X}$$



Handling missing values at test time



Given a partial observation $\mathbf{x}^o$, what is the expected output from f?

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

Khosravi et al., "On Tractable Computation of Expected Predictions", 2019

Fairness analysis

```
using ProbabilisticCircuits
pc = load_prob_circuit(zoo_psdd_file("insurance.psdd"));
rc = load_logistic_circuit(zoo_lc_file("insurance.circuit"), 1);
```

q: *Is the predictive model biased by gender?*

```
groups = make_observations([["male"], ["female"]])
exps, _ = Expectation(pc, rc, groups);
println("Female  : \$ $(exps[2])");
println("Male    : \$ $(exps[1])");
println("Diff    : \$ $(exps[2] - exps[1])");
Female  : $ 14170.125469335406
Male    : $ 13196.548926381849
Diff    : $ 973.576542953568
```

---

https://github.com/Juice-jl/

Causal Inference

Given subsets $\boldsymbol{A}, \boldsymbol{Y} \subseteq \boldsymbol{X}$, interested in *causal effect* $p(\boldsymbol{Y}|do(\boldsymbol{A}))$.

Causal Inference

Given subsets $\boldsymbol{A}, \boldsymbol{Y} \subseteq \boldsymbol{X}$, interested in *causal effect* $p(\boldsymbol{Y}|do(\boldsymbol{A}))$.

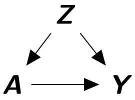
In general, $p(\boldsymbol{Y}|do(\boldsymbol{A})) \neq p(\boldsymbol{Y}|\boldsymbol{A})$ (correlation is not causation).

Causal Inference

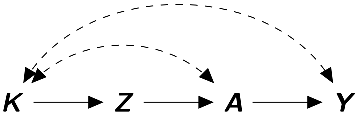
Given subsets $\boldsymbol{A}, \boldsymbol{Y} \subseteq \boldsymbol{X}$, interested in *causal effect* $p(\boldsymbol{Y}|do(\boldsymbol{A}))$.

In general, $p(\boldsymbol{Y}|do(\boldsymbol{A})) \neq p(\boldsymbol{Y}|\boldsymbol{A})$ (correlation is not causation).

- Specify (qualitative) assumptions on the system using a **causal diagram** G (here $\boldsymbol{A}, \boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{K} \subseteq \boldsymbol{X}$)) :



(a) Backdoor



(b) Napkin

Causal Inference

Given subsets $\boldsymbol{A}, \boldsymbol{Y} \subseteq \boldsymbol{X}$, interested in *causal effect* $p(\boldsymbol{Y}|do(\boldsymbol{A}))$.
In general, $p(\boldsymbol{Y}|do(\boldsymbol{A})) \neq p(\boldsymbol{Y}|\boldsymbol{A})$ (correlation is not causation).

- Specify (qualitative) assumptions on the system using a **causal diagram** G (here $\boldsymbol{A}, \boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{K} \subseteq \boldsymbol{X}$)) :

$$\begin{tikzcd} & \boldsymbol{Z} \arrow[dl] \arrow[dr] & \\ \boldsymbol{A} \arrow[rr] & & \boldsymbol{Y} \end{tikzcd}$$

(a) Backdoor

$$\begin{tikzcd} \boldsymbol{K} \arrow[r] \arrow[rr, bend left, dashed, leftrightarrow] \arrow[rrr, bend left=50, dashed, leftrightarrow] & \boldsymbol{Z} \arrow[r] & \boldsymbol{A} \arrow[r] & \boldsymbol{Y} \end{tikzcd}$$

(b) Napkin

- Given causal diagram G, can derive expressions for causal effect $p(\boldsymbol{Y}|\boldsymbol{A})$ using *do-calculus* (Pearl 1995).

$$\sum_{\boldsymbol{Z}} p(\boldsymbol{Z}) p(\boldsymbol{Y}|\boldsymbol{A}, \boldsymbol{Z})$$

(a) Backdoor

$$\frac{\sum_{\boldsymbol{K}} p(\boldsymbol{A}, \boldsymbol{Y}|\boldsymbol{K}, \boldsymbol{Z}) p(\boldsymbol{K})}{\sum_{\boldsymbol{K}} p(\boldsymbol{A}|\boldsymbol{K}, \boldsymbol{Z}) p(\boldsymbol{K})}$$

(b) Napkin

Tractability of Exact Causal Inference

Consider the backdoor query, for **fixed** values of the treatment $\boldsymbol{a}$ and outcome $\boldsymbol{y}$:

$$p(\boldsymbol{y}|do(\boldsymbol{a})) := \sum_{\boldsymbol{Z}} p(\boldsymbol{Z}) \times p(\boldsymbol{y}|\boldsymbol{a}, \boldsymbol{Z})$$

.

Tractability of Exact Causal Inference

Consider the backdoor query, for **fixed** values of the treatment $\boldsymbol{a}$ and outcome $\boldsymbol{y}$:

$$p(\boldsymbol{y}|do(\boldsymbol{a})) := \sum_{\boldsymbol{Z}} p(\boldsymbol{Z}) \times p(\boldsymbol{y}|\boldsymbol{a}, \boldsymbol{Z})$$

.

Theorem (Wang & Kwiatkowska 2023)

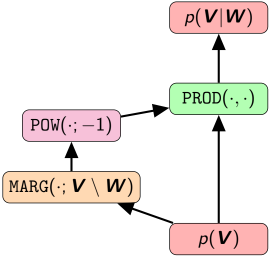
If p is given as a structured decomposable and deterministic circuit, then the backdoor query is #P-hard to compute.

Applying the Atlas of Tractable Operations

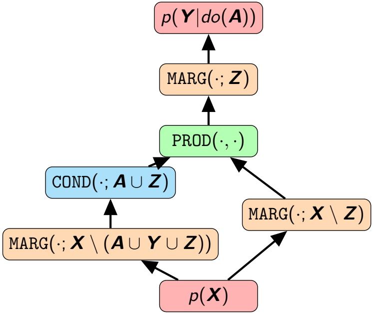
Break down do-calculus query into compositions of basic operations, such as marginalization, products, and powers:

Applying the Atlas of Tractable Operations

Break down do-calculus query into compositions of basic operations, such as marginalization, products, and powers:



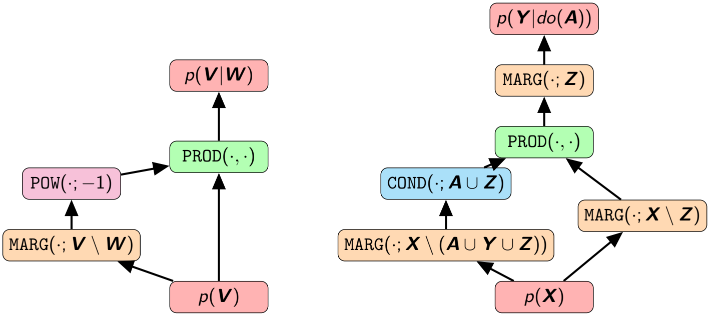
(a) Pipeline for $\texttt{COND}(\cdot,\boldsymbol{W})$



(b) Pipeline for entire backdoor query

Applying the Atlas of Tractable Operations

Break down do-calculus query into compositions of basic operations, such as marginalization, products, and powers:



(a) Pipeline for $\texttt{COND}(\cdot,\boldsymbol{W})$ (b) Pipeline for entire backdoor query

Problem: Cannot guarantee that input to $\texttt{POW}$ is deterministic, even if $p(\boldsymbol{X})$ is deterministic.

Marginal Determinism

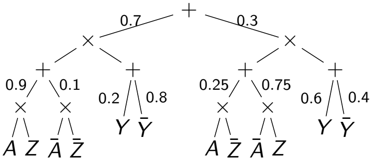
Definition (Marginal Determinism, Choi et al. 2020)

Given a subset of variables $\boldsymbol{Q} \subseteq \boldsymbol{X}$, a PC is $\boldsymbol{Q}$-deterministic if the children of a sum node T correspond to different values of $\boldsymbol{Q}$ (for sum nodes with $sc(T) \cap \boldsymbol{Q} \neq \emptyset$).

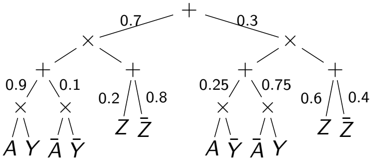
Marginal Determinism

Definition (Marginal Determinism, Choi et al. 2020)

Given a subset of variables $\boldsymbol{Q} \subseteq \boldsymbol{X}$, a PC is $\boldsymbol{Q}$-deterministic if the children of a sum node T correspond to different values of $\boldsymbol{Q}$ (for sum nodes with $sc(T) \cap \boldsymbol{Q} \neq \emptyset$).



(a) $\boldsymbol{Q} = \{A, Z\}$-deterministic

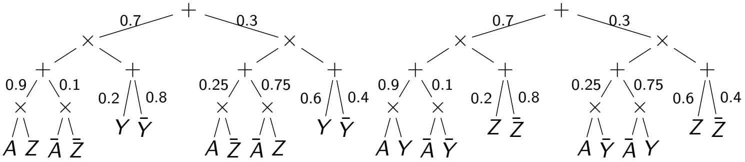


(b) $\boldsymbol{Q} = \{A, Y\}$-deterministic

Marginal Determinism

Definition (Marginal Determinism, Choi et al. 2020)

Given a subset of variables $\boldsymbol{Q} \subseteq \boldsymbol{X}$, a PC is $\boldsymbol{Q}$-deterministic if the children of a sum node T correspond to different values of $\boldsymbol{Q}$ (for sum nodes with $sc(T) \cap \boldsymbol{Q} \neq \emptyset$).



(a) $\boldsymbol{Q} = \{A, Z\}$-deterministic (b) $\boldsymbol{Q} = \{A, Y\}$-deterministic

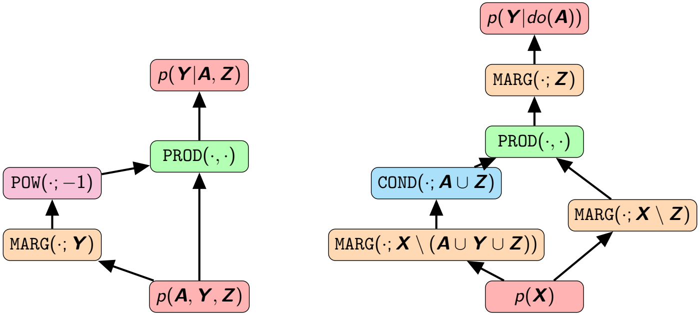
Motivation: If a circuit is marginally deterministic w.r.t $\boldsymbol{Q}$, then we can marginalize out $\boldsymbol{X} \setminus \boldsymbol{Q}$ and obtain a deterministic circuit!

Tractable Causal Inference

If (the circuit encoding) $p(\boldsymbol{X})$ is $(\boldsymbol{A} \cup \boldsymbol{Z})$-deterministic, then the input to `POW` is guaranteed to be deterministic.

Tractable Causal Inference

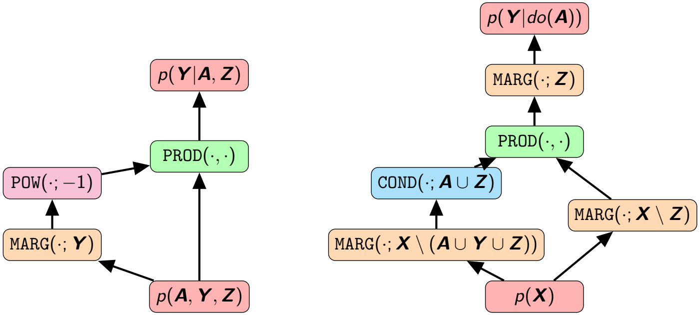
If (the circuit encoding) $p(\boldsymbol{X})$ is $(\boldsymbol{A} \cup \boldsymbol{Z})$-deterministic, then the input to `POW` is guaranteed to be deterministic.



(a) Pipeline for $\mathtt{COND}(\cdot, \boldsymbol{A} \cup \boldsymbol{Z})$ (b) Pipeline for entire backdoor query

Tractable Causal Inference

If (the circuit encoding) $p(\boldsymbol{X})$ is $(\boldsymbol{A} \cup \boldsymbol{Z})$-deterministic, then the input to POW is guaranteed to be deterministic.

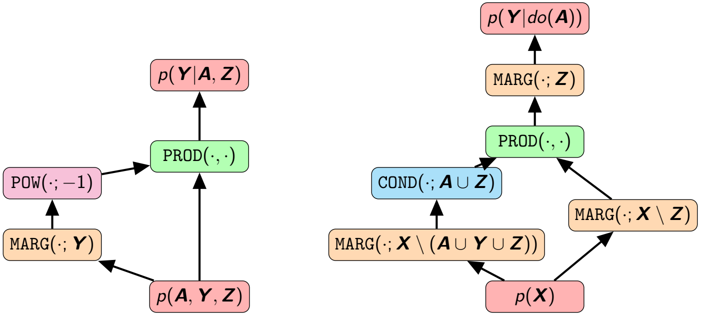


(a) Pipeline for COND$(\cdot,\boldsymbol{A}\cup\boldsymbol{Z})$ (b) Pipeline for entire backdoor query

$\Longrightarrow$ all operations are tractable according to Atlas

Tractable Causal Inference

If (the circuit encoding) $p(\boldsymbol{X})$ is $(\boldsymbol{A} \cup \boldsymbol{Z})$-deterministic, then the input to POW is guaranteed to be deterministic.



(a) Pipeline for COND$(\cdot, \boldsymbol{A} \cup \boldsymbol{Z})$ (b) Pipeline for entire backdoor query

$\implies$ all operations are tractable according to Atlas

$\implies$ can compute causal effect in $O(|p|^3)$ time

(can improve to $O(|p|^2)$)

Open Questions

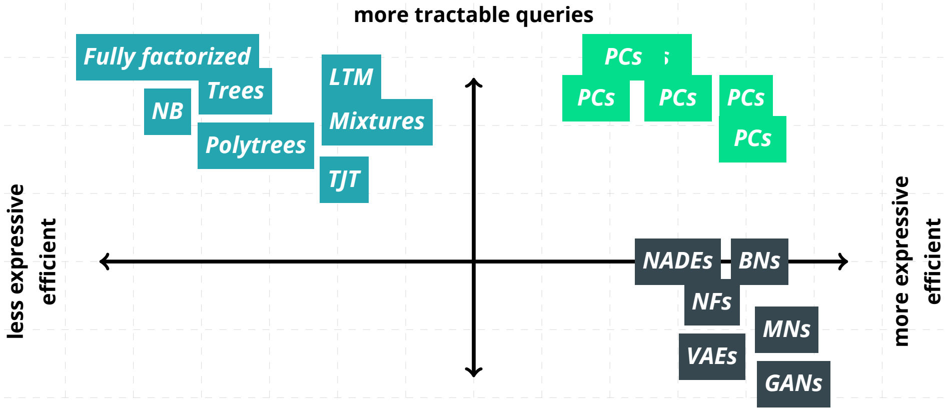
- Are all causal queries derived by the do-calculus tractable in `PTIME` (for some non-trivial marginal determinism condition)?
- What is the optimal complexity for these queries?

Questions answered today

1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. ***Are all tractable distributions probabilistic circuits?*** (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

Questions answered today

1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. ***Are all tractable distributions probabilistic circuits?*** (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)



tractability vs expressive efficiency

Smooth ∨ decomposable ∨ deterministic ∨ structured decomposable PCs?

	smooth	*dec.*	det.	str.dec.
Arithmetic Circuits (ACs) *(Darwiche 2003)*	✔	✔	✔	✘
Sum-Product Networks (SPNs) *(Poon et al. 2011)*	✔	✔	✘	✘
Cutset Networks (CNets) *(Rahman et al. 2014)*	✔	✔	✔	✘
Probabilistic Decision Graphs *(Jaeger 2004)*	✔	✔	✔	✔
(Affine) ADDs *(Hoey et al. 1999; Sanner et al. 2005)*	✔	✔	✔	✔
AndOrGraphs *(Dechter et al. 2007)*	✔	✔	✔	✔
PSDDs *(Kisa et al. 2014)*	✔	✔	✔	✔

Low-treewidh PGMs

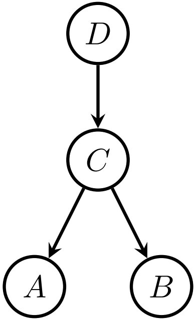
Tree, polytrees and
Thin Junction trees
can be turned into

- decomposable
- smooth
- deterministic

circuits

Therefore they support
tractable

- EVI
- MAR/CON
- MAP



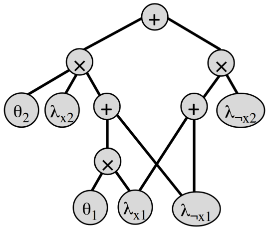
Arithmetic Circuits (ACs)

ACs *(Darwiche 2003)* are

- decomposable
- smooth
- deterministic

They support tractable

- EVI
- MAR/CON
- MAP



$\Rightarrow$ *parameters are attached to the leaves*

$\Rightarrow$ *...but can be moved to the sum node edges (Rooshenas et al. 2014)*

Lowd and Rooshenas, "Learning Markov Networks With Arithmetic Circuits", 2013

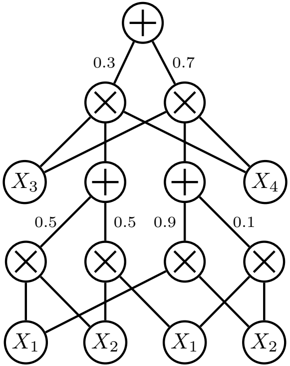
Sum-Product Networks (SPNs)

SPNs *(Poon et al. 2011)* are

- decomposable
- smooth
- ~~deterministic~~

They support tractable

- EVI
- MAR/CON
- ~~MAP~~



$\Longrightarrow$ *deterministic SPNs are also called selective* *(Peharz et al. 2014)*

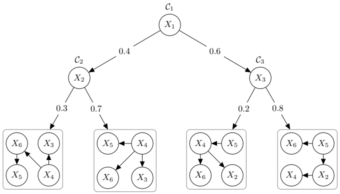
Cutset Networks (CNets)

CNets
(Rahman et al. 2014) are

- decomposable
- smooth
- deterministic

They support tractable

- EVI
- MAR/CON
- MAP



Rahman et al., "Cutset Networks: A Simple, Tractable, and Scalable Approach for Improving the Accuracy of Chow-Liu Trees", 2014
Di Mauro et al., "Learning Accurate Cutset Networks by Exploiting Decomposability", 2015

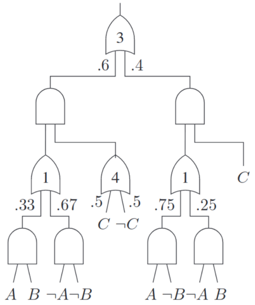
Probabilistic Sentential Decision Diagrams

PSDDs *(Kisa et al. 2014)* are

- structured decomposable
- smooth
- deterministic

They support tractable

- EVI
- MAR/CON
- MAP
- Complex queries!



Kisa et al., "Probabilistic sentential decision diagrams", 2014
Choi et al., "Tractable learning for structured probability spaces: A case study in learning preference distributions", 2015
Shen et al., "Conditional PSDDs: Modeling and learning with modular knowledge", 2018

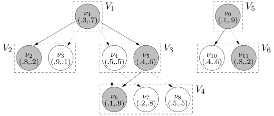
Probabilistic Decision Graphs

PDGs *(Jaeger 2004)* are

- structured decomposable
- smooth
- deterministic

They support tractable

- EVI
- MAR/CON
- MAP
- Complex queries!



Jaeger, "Probabilistic decision graphs—combining verification and AI techniques for probabilistic inference", 2004
Jaeger et al., "Learning probabilistic decision graphs", 2006

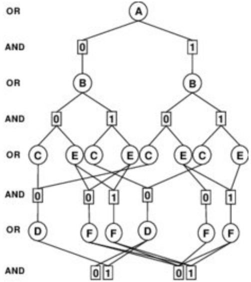
AndOrGraphs

AndOrGarphs
(Dechter et al. 2007) are

- structured decomposable
- smooth
- deterministic

They support tractable

- EVI
- MAR/CON
- MAP
- Complex queries!



Dechter and Mateescu, "AND/OR search spaces for graphical models", 2007
Marinescu and Dechter, "Best-first AND/OR search for 0/1 integer programming", 2007

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

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

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

Enter: Determinantal Point Processes (DPPs)

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

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

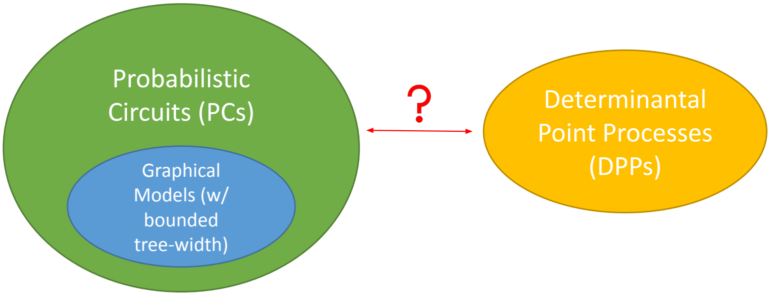
Tractable likelihoods and marginals

Global Negative Dependence

Diversity in recommendation systems

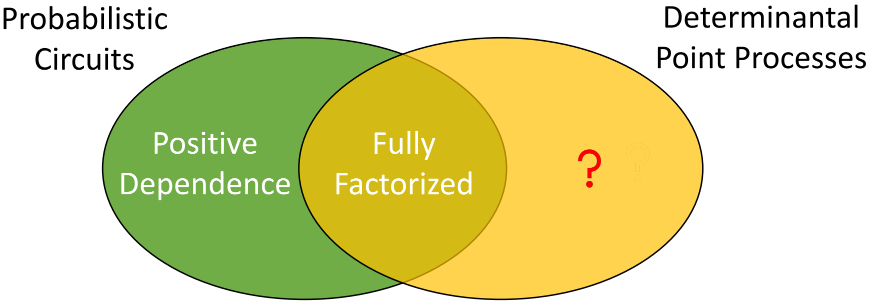
$$\Pr_L(X_1 = 1, X_2 = 0, X_3 = 1, X_4 = 0) = \frac{1}{\det(L + I)} \det(L_{\{1,2\}})$$

Are all tractable probabilistic models probabilistic circuits?



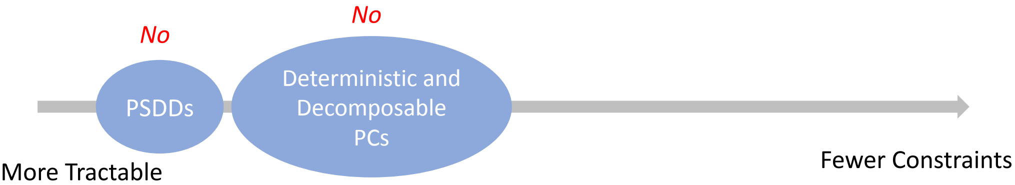
Honghua Zhang, Steven Holtzen and Guy Van den Broeck. On the Relationship Between Probabilistic Circuits and Determinantal Point Processes, *UAI*, 2020.

Relationship between PCs and DPPs



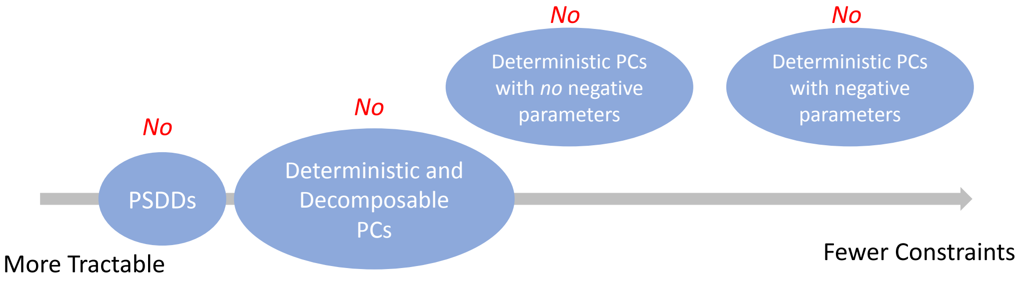
Honghua Zhang, Steven Holtzen and Guy Van den Broeck. On the Relationship Between Probabilistic Circuits and Determinantal Point Processes, *UAI*, 2020.

We cannot tractably represent DPPs with subclasses of PCs



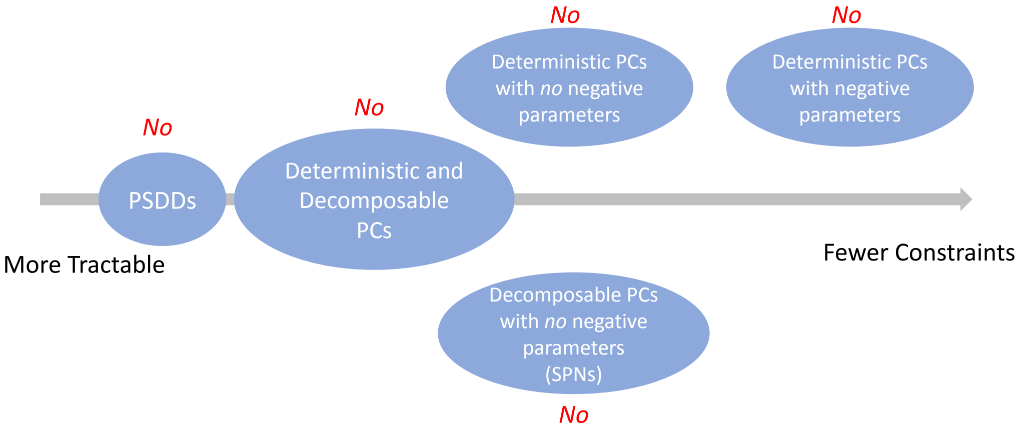
Honghua Zhang, Steven Holtzen and Guy Van den Broeck. On the Relationship Between Probabilistic Circuits and Determinantal Point Processes, *UAI*, 2020.

We cannot tractably represent DPPs with subclasses of PCs



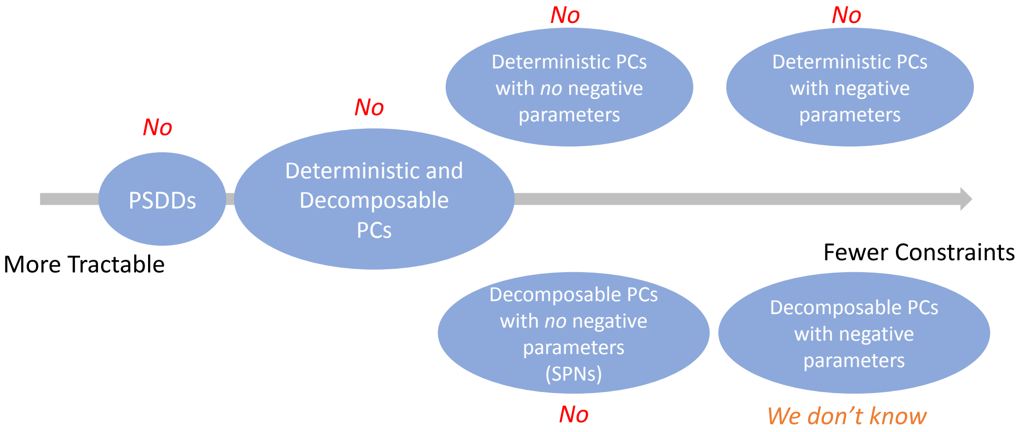
Honghua Zhang, Steven Holtzen and Guy Van den Broeck. On the Relationship Between Probabilistic Circuits and Determinantal Point Processes, *UAI*, 2020.

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.

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.

PCs and Circuit Lower Bounds

Theorem (Martens and Medabalimi, 2014). *Let P_n be the uniform distribution over spanning trees on K_n. For $n \geq 20$, the size of any smooth and decomposable PC that represents P_n is at least $2^{n/30240}$.*

Based on arithmetic circuit lower bounds by Ran Raz and Amir Yehudayoff

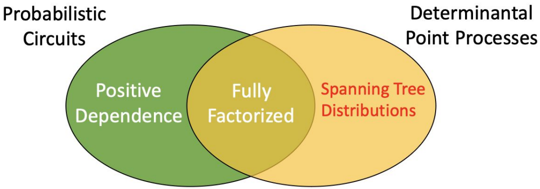
Decomposable PCs are **Syntactically Multilinear** Arithmetic Circuits:

Definition 7 (Multilinear Arithmetic Circuit) If every node of an arithmetic circuit Φ over y computes a multilinear polynomial in y, Φ is said to be a *(semantically) multilinear arithmetic circuit*. And if for every product node in Φ, the scopes of its child nodes are pair-wise disjoint, Φ is said to be a *syntactically multilinear arithmetic circuit*.

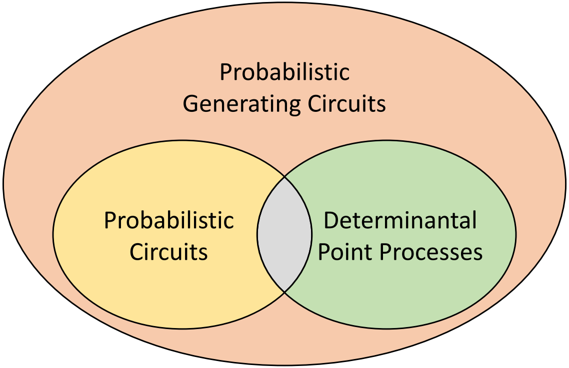
DPPs have No Compact Decomposable PCs

Theorem (Snell, 1995). *The uniform distribution over spanning trees on the complete graph K_n is a DPP over $\binom{n}{2}$ edges.*

Theorem (Martens and Medabalimi, 2014). *Let P_n be the uniform distribution over spanning trees on K_n. For $n \geq 20$, the size of any smooth and decomposable PC that represents P_n is at least $2^{n/30240}$.*



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

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

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

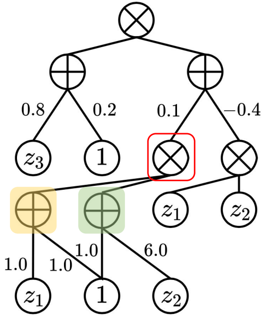
$$g_\beta = \boxed{0.16 z_1 z_2 z_3} + 0.04 z_1 z_2 + 0.08 z_1 z_3 + 0.02 z_1 + 0.48 z_2 z_3 + 0.12 z_2 + 0.08 z_3 + 0.02.$$

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

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.

PCs as PGCs

(Smooth & Decomposable) PCs represents probability mass functions:

$$m_\beta = 0.16X_1X_2X_3 + 0.04X_1X_2\overline{X_3} + 0.08X_1\overline{X_2}X_3 + 0.02X_1\overline{X_2}\,\overline{X_3}$$
$$+0.48\overline{X_1}X_2X_3 + 0.12\overline{X_1}X_2\overline{X_3} + 0.08\overline{X_1}\,\overline{X_2}X_3 + 0.02\overline{X_1}\,\overline{X_2}\,\overline{X_3}$$

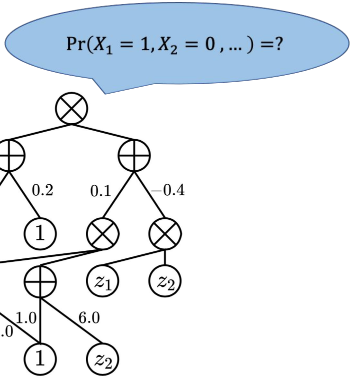
PGCs represent probability generating functions:

$$g_\beta = 0.16z_1z_2z_3 + 0.04z_1z_2 + 0.08z_1z_3 + 0.02z_1$$
$$+0.48z_2z_3 + 0.12z_2 + 0.08z_1z_3 + 0.02$$

Given a smooth & decomposable PC, by setting $\overline{X_i}$ to 1, and X_i to z_i, we obtain a PGC that represents the PC.

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

Tractable Likelihood (EVID) or Marginals (MAR)?



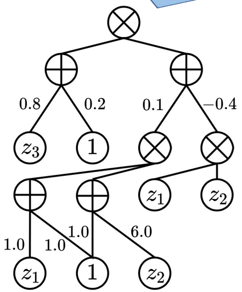
Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

PGCs Support Tractable Likelihoods/Marginals

Purely symbolic

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

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



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

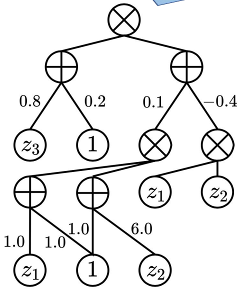
Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

PGCs Support Tractable Likelihoods/Marginals

Purely symbolic

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

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



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

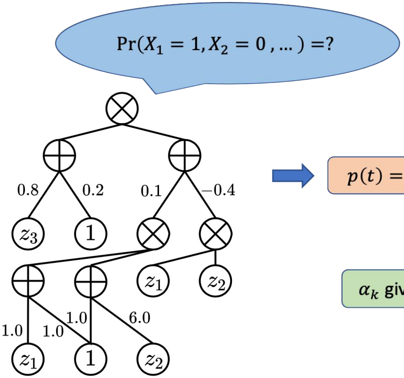
- Monomials setting to true variables that must be false are 0-ed out
- Other monomials contribute to result.
- Only monomials that set all required variables to true have max degree.
- Sum those up

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

PGCs Support Tractable Likelihoods/Marginals

Purely symbolic

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



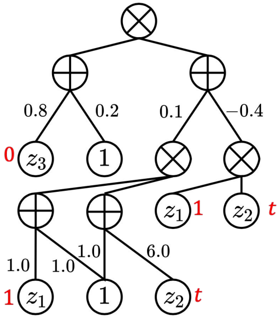
$p(t) = \alpha_k t^k + \cdots + \alpha_1 t$

α_k gives the answer

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

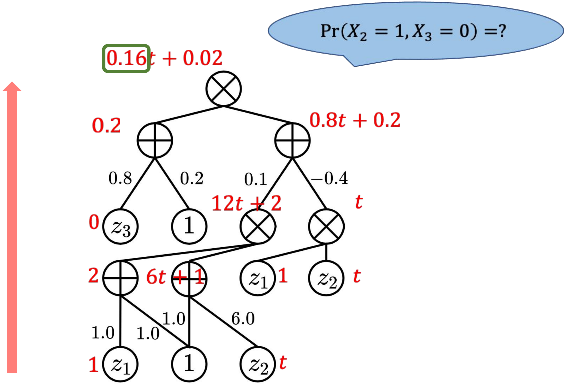
Example

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



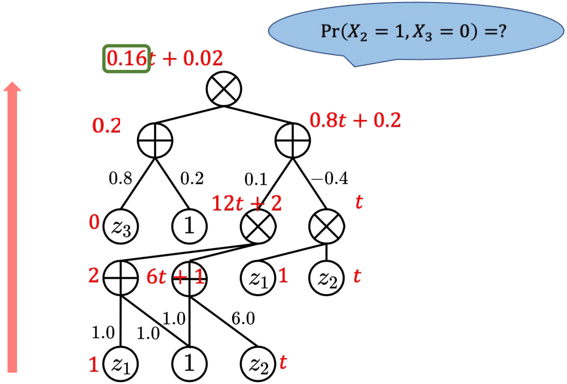
Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

Example



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

Example



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.

Inference Time Complexity

Given a PGC of size *m* (#edges) over *n* random variables.

Algorithm 1 (*Zhang et al., ICML 2021*):

Bottom-up pass w/ z_i = t, 0 or 1	+	Product/sum of degree-n polynomials at each node	=	$O(mn^2)$

or O(mn log n log log n)

Inference Time Complexity

Given a PGC of size *m* (#edges) over *n* random variables.

Algorithm 1 (*Zhang et al., ICML 2021*):

Bottom-up pass w/ z_i = t, 0 or 1	+	Product/sum of degree-n polynomials at each node	=	$O(mn^2)$

or O(mn log n log log n)

Algorithm 2 (*Harviainen et al., UAI 2023*):

observation: the output of a PGC is a degree-n polynomial w/ respect to t

Bottom-up pass w/ t = 0, 1, …, n	+	Polynomial interpolation at t = 0, 1, …, n	=	$O(mn)$

Syntactic vs. Semantic Restrictions

+ PGCs are tractable when semantically multilinear
  + No need for PC decomposability/syntactic multilinearity or other properties…
- Checking Validity of PGCs is Hard

> Theorem (*Harviainen et al.*). It is NP-hard to check if a PGC encodes a valid probability generating polynomial

DPPs as PGCs

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

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

We need it as a sum of products to obtain a Probabilistic Generating Circuit

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:

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

Constant

We need it as a sum of products to obtain a Probabilistic Generating Circuit

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.

Experiment Results: Amazon Baby Registries

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

SimplePGC achieves SOTA result on 11/15 datasets

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

Beyond DPPs: Strongly Rayleigh Distributions

DPPs are strongly Rayleigh distributions

Definition. *A probability distribution over binary random variables* $X_1, \ldots, X_n$ *(or equivalently, subsets of* $[n] := \{1, 2, \ldots, n\}$*) is strongly Rayleigh if its probability generating polynomial* g *is real-stable; that is, for* $z_i \in \mathbb{C}$*, if* $Im(z_i) > 0$ *for all* z_i*, then* $g(z_1, \cdots, z_n) \neq 0$.

We can efficiently sample from strongly Rayleigh distributions by MCMC (with polynomial bound on mixing time)

Efficient Sampling from SR Distributions

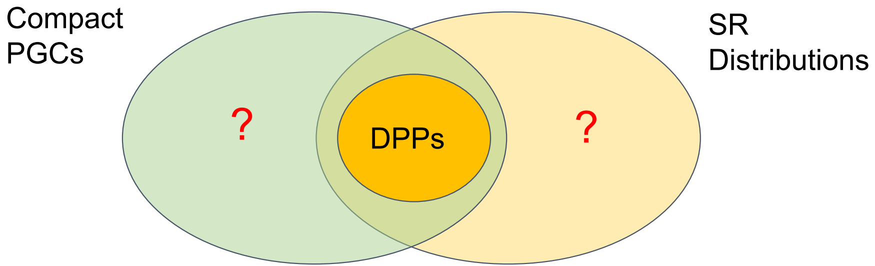
Theorem (Li et al., 2016). *Let π be a strongly Rayleigh distribution over $[n]$, we can efficiently sample from π by sampling from its symmetric homogenization π_{sh}; for $S \subset [2n]$, define*

$$\pi_{sh}(S) := \begin{cases} \pi(S \cap [n]) \binom{n}{S \cap [n]}^{-1}, & \text{if } |S| = n \\ 0, & \text{otherwise} \end{cases}$$

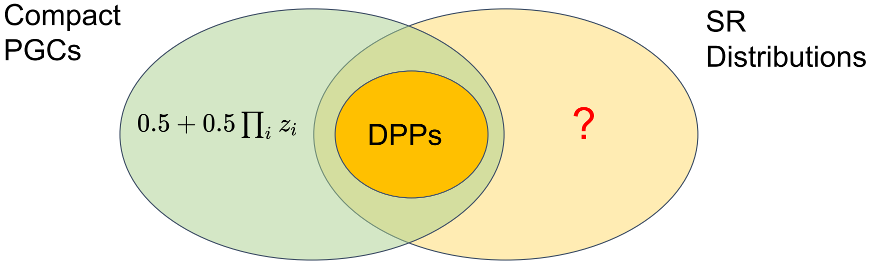
in particular, π_{sh} is also strongly Rayleigh and the mixing time of a Gibbs-exchange sampler with initial set S_0 is bounded as

$$\tau(\epsilon) \leq 2n^2 \left(\log \binom{n}{|S_0|} + \log \pi(S_0)^{-1} + \log \epsilon^{-1} \right)$$

Relationship between PGCs and SR Distributions



Relationship between PGCs and SR Distributions



Not All SR Distributions have Compact PGCs (Bläser 2023)

Let $K_{m,n} = (U \cup V, E)$ be a complete bipartite graph, the signed double function generating polynomial is defined as

$$DF_{m,n}(e) = \sum_{F,H} (-1)^{|F|+|H|} \prod_{(i,j)\in F} e_{i,j} \prod_{(i',j')\in H} e_{i',j'}$$

where the sum is taken over all partial functions $U \to V$ and $V \to U$, respectively. Each pair of (F, H) is a double function of $K_{m,n}$.

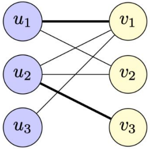


Figure 4. The thick edges are a matching of size two.

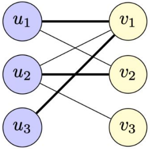


Figure 5. The thick edges form a total function $U \to V$, which is not injective.

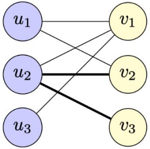


Figure 6. The thick edges form a partial function from V to U.

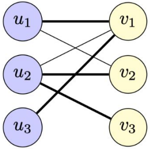


Figure 7. A double function.

Not All SR Distributions have Compact PGCs (Bläser 2023)

$$DF_{m,n}(e) = \sum_{F,H} (-1)^{|F|+|H|} \prod_{(i,j)\in F} e_{i,j} \prod_{(i',j')\in H} e_{i',j'}$$

Generalize to bipartite <u>multigraph</u> $K_{m,n}^{(d)}$

d: each edge from U to V has d copies

$$DF_{m,n}^{(d)}(e^{(d)}) = \sum_{F,H} (-1)^{|F|+|H|} \prod_{(i,j)\in F\setminus H} \sum_{\delta=1}^{d} e_{i,j}^{(\delta)} \prod_{(i',j')\in H\cap F} \sum_{1\le\delta'<\gamma\le d} e_{i',j'}^{(\delta')} e_{i',j'}^{(\gamma)} \prod_{(i'',j'')\in H\setminus F} \sum_{\delta''=1}^{d} e_{i'',j''}^{(\delta'')}$$

$DF_{n,n}^{(n+2)}$ is real-stable and its evaluation is #P-hard.

$DF_{n,n}^{(n+2)}$ *does not* define an SR distribution as it has negative coefficients

Not All SR Distributions have Compact PGCs (Bläser 2023)

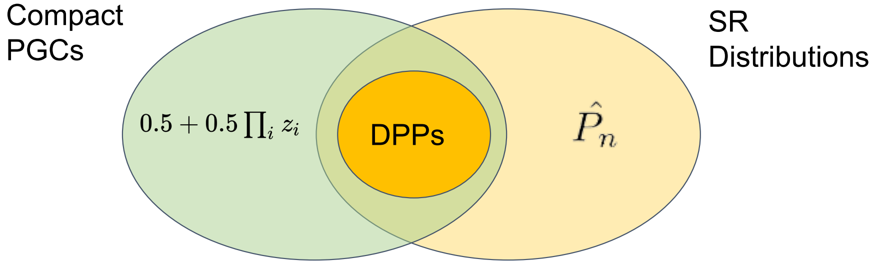
Definition. *For a polynomial $f(z_1, \ldots, z_n)$ with z_i of degree k_i, the inversion of f is defined as $\prod_i {z_i}^{k_i} f(-1/z_1, \ldots, -1/z_i, \ldots, -1/z_n)$.*

The inversion of a real stable polynomial is also real stable

Let P_n be the inversion of $DF_{n,n}^{(n+2)}$, then P_n is a mutilinear and real stable polynomial with all coefficients non-negative.

Theorem (Bläser, 2023). *Assuming $P^{\#P} \not\subseteq P/Poly$. Let $\hat{P}_n$ be the normalized P_n, then $\hat{P}_n$ cannot be represented as polynomial-size PGCs.*

Relationship between PGCs and SR Distributions



Probabilistic **generating** circuits seem awfully general.

*Are all tractable probabilistic models probabilistic **generating** circuits?*

Questions answered today

1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. ***Are all tractable distributions probabilistic circuits?*** (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

Questions answered today

1. *What are probabilistic queries? Are current models tractable?* (Guy)
2. *What are probabilistic circuits and why are they tractable?* (Guy)
3. *What is the connection to logical circuit languages?* (YooJung)
4. *How do I compile my favorite model into a circuit?* (YooJung)
5. *How are circuit size and tractability related?* (YooJung)
6. *What's the most impressive query we can efficiently compute?* (YooJung)
7. ***Are all tractable distributions probabilistic circuits?*** (Guy)
8. *How to learn probabilistic circuits from data?* (Guy)

Building Probabilistic Circuits

Information

Prior Knowledge
domain assumptions
constraints
other models

Data
experimental data
samples
measurements

compilation

learning

Circuits

decomposability
smoothness
determinism
compatibility

Structure

Parameters

$\boldsymbol{\theta}, \boldsymbol{w}$

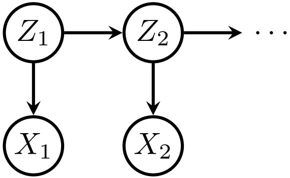
generative
discriminative
Bayesian
credal

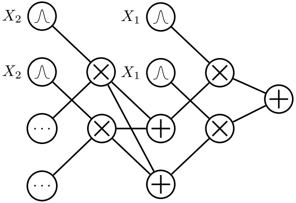
Origins: Compilation

Compiling probabilistic graphical models

Arithmetic circuits *(Darwiche 2002, 2003, 2009)*

- Compile a given Bayesian network into an **arithmetic circuit**—a smooth, decomposable and deterministic PCs
- Either via logic encoding of Bayesian network + knowledge compilation
- Or record "execution trace" (sum and product operations) of traditional inference algorithms (junction tree, variable elimination)





Compilation

Selected references

Logic circuits, interplay between structural properties and tractable reasoning

(Darwiche et al. 2002a)

Converting probabilistic graphical models via knowledge compilation

(Darwiche 2002)

Logic circuit compilers

(Darwiche 2004; Muise et al. 2012; Bova et al. 2015; Lagniez et al. 2017; Oztok et al. 2018)

Neuro-symbolic models using logic circuits

(Ahmed et al. 2022a,b)

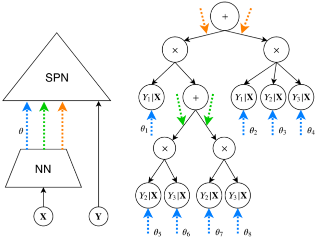
Parameter Learning

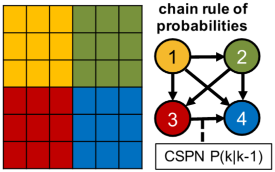
Gradient descent (of course)

- PCs are computational graphs
- Hence we can just learn them as any other neural net using SGD
- Use re-parameterization if parameters should satisfy constraints:
  - soft-max for sum-weights (non-negative, sum-to-one)
  - soft-plus for variances
  - low-rank plus diagonal for covariance matrices
- Allows for conditional distributions

Conditional PCs

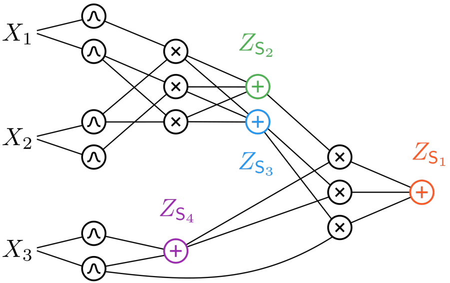
(Shao et al. 2019)

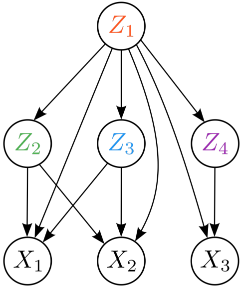




Maximum likelihood (frequentist)

PCs can be interpreted as ***hierarchical latent variable models***, where each sum node corresponds to a discrete latent variable *(Peharz et al. 2016)*. This allows to perform **classical maximum-likelihood** estimation.





Closed-form maximum likelihood

When the circuit is **deterministic**, there is even an ***closed-form ML solution***, which is incredible fast:

```
julia> using ProbabilisticCircuits;
julia> data, structure = load(...);
julia> num_examples(data)
17412
julia> num_edges(structure)
270448
julia> @btime estimate_parameters(structure, data);
  63.585 ms (1182350 allocations: 65.97 MiB)
```

Custom SIMD and CUDA kernels to parallelize over layers and training examples.

https://github.com/Juice-jl/

Expectation-Maximization

When the PC is not deterministic, we can still apply ***expectation-maximization*** *(Peharz et al. 2016).* EM can piggy-back on autodfiff:

```
train_x, valid_x, test_x = get_mnist_images([7])

graph = Graph.poon_domingos_structure(shape=(28,28), delta=[7])
args = EinsumNetwork.Args(num_var=train_x.shape[1], num_dims=1,
                          num_classes=1, num_sums=28,
                          num_input_distributions=28,
                          exponential_family=EinsumNetwork.BinomialArray,
                          exponential_family_args={'N':255},
                          online_em_frequency=1, online_em_stepsize=0.05)

PC = EinsumNetwork.EinsumNetwork(graph, args)
PC.initialize()
PC.to('cuda')
```

---

https://github.com/cambridge-mlg/EinsumNetworks

Expectation-Maximization

```
for epoch_count in range(10):
    train_ll, valid_ll, test_ll = compute_loglikelihood()
    start_t = time.time()

    for idx in get_batches(train_x, 100):
        outputs = PC.forward(train_x[idx, :])
        log_likelihood = EinsumNetwork.log_likelihoods(outputs).sum()
        log_likelihood.backward()
        PC.em_process_batch()

    print_performance(epoch_count, train_ll, valid_ll, test_ll, time.time() - start_t)
```

https://github.com/cambridge-mlg/EinsumNetworks

Expectation-Maximization

```
# train sample: 5175
# parameters: 1573486

[epoch 0]   train LL -140936.80   valid LL -140955.72   test LL -141033.80   ... elapsed time 3.621 sec
[epoch 1]   train LL  -15916.14   valid LL  -15693.25   test LL  -15976.43   ... elapsed time 3.438 sec
[epoch 2]   train LL  -10865.67   valid LL  -10616.72   test LL  -10943.56   ... elapsed time 3.436 sec
[epoch 3]   train LL  -10388.53   valid LL  -10158.84   test LL  -10475.49   ... elapsed time 3.473 sec
[epoch 4]   train LL  -10264.11   valid LL  -10041.66   test LL  -10352.59   ... elapsed time 3.497 sec
[epoch 5]   train LL  -10212.66   valid LL  -10001.09   test LL  -10319.35   ... elapsed time 3.584 sec
[epoch 6]   train LL  -10192.21   valid LL   -9965.98   test LL  -10314.84   ... elapsed time 3.508 sec
[epoch 7]   train LL  -10153.97   valid LL   -9920.09   test LL  -10261.41   ... elapsed time 3.446 sec
[epoch 8]   train LL  -10112.95   valid LL   -9882.48   test LL  -10236.34   ... elapsed time 3.579 sec
[epoch 9]   train LL  -10093.31   valid LL   -9862.15   test LL  -10200.94   ... elapsed time 3.483 sec
```

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

Structure Learning

Region graphs

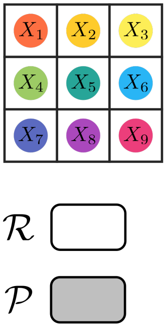
Laying out the PC structure on a high level

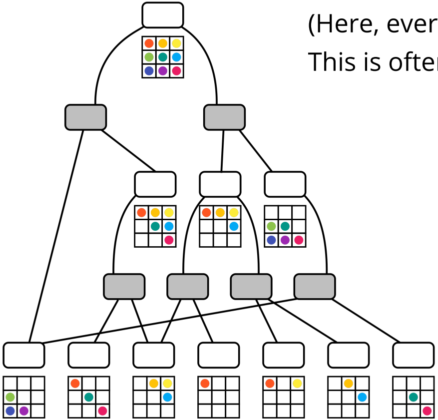
- Region graphs (RGs) describe decompositional structure
- RGs are *bipartite, directed graphs* containing *regions* ($\mathcal{R}$) and *partitions* ($\mathcal{P}$)
- Input and output nodes of the RG are regions
- Regions have a *scope* (receptive field), denoted as $sc(\mathcal{R}) \subseteq \mathbf{X}$
- For every partition $\mathcal{P}$ it holds that

$$sc(\mathcal{R}_{out}) = \bigcup_{\mathcal{R}_{in} \in inputs(\mathcal{P})} sc(\mathcal{R}_{in})$$

$$sc(\mathcal{R}') \cap sc(\mathcal{R}'') = \emptyset, \qquad \mathcal{R}' \neq \mathcal{R}'' \in inputs(\mathcal{P})$$

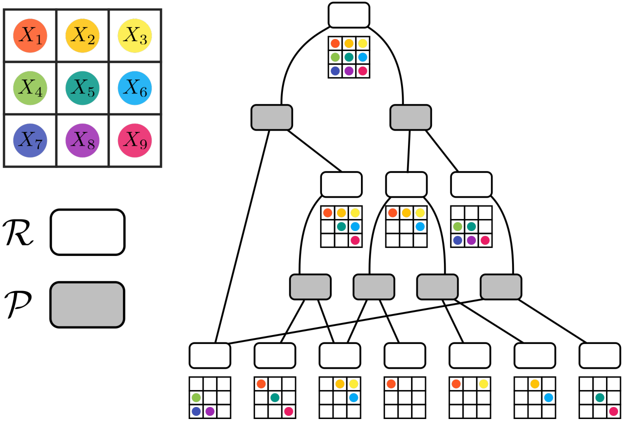
Example region graph





(Here, every partition has 2 input regions.
This is often assumed, but not necessary.)

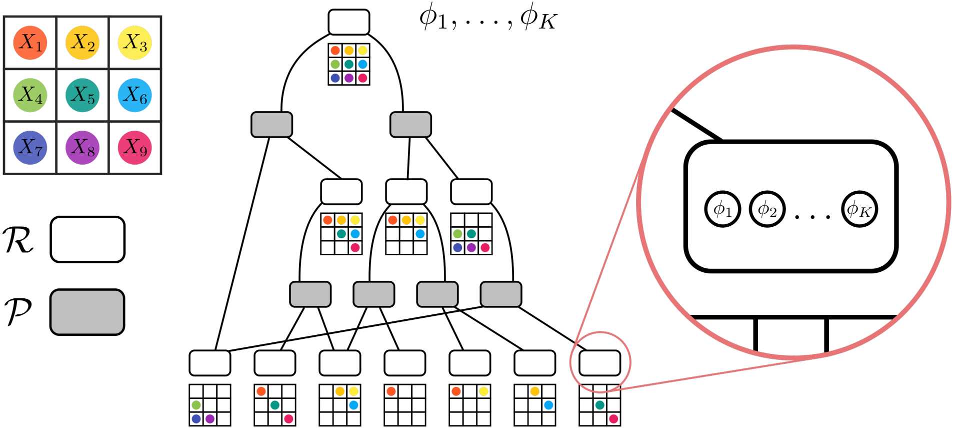
From region graphs to PCs



From region graphs to PCs

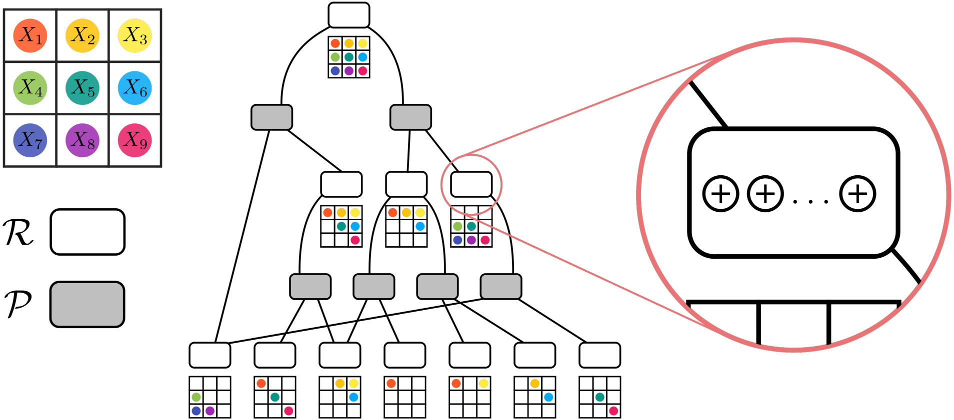
Equip each input region with non-linear units

$\phi_1, \ldots, \phi_K$



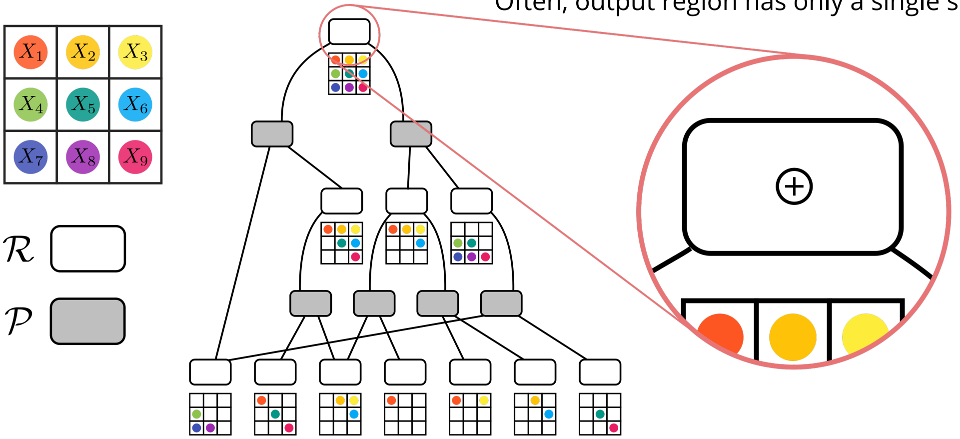
From region graphs to PCs

Equip each internal region with sum nodes



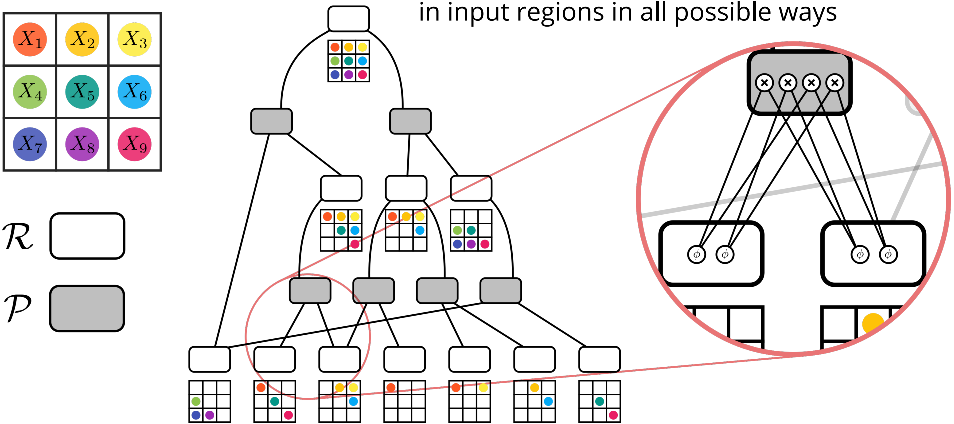
From region graphs to PCs

Often, output region has only a single sum



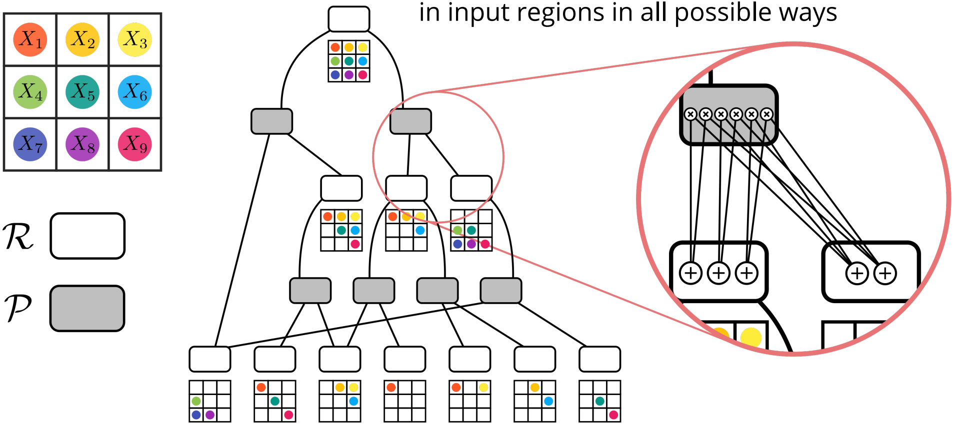
From region graphs to PCs

Equip partitions with products, combining units in input regions in all possible ways



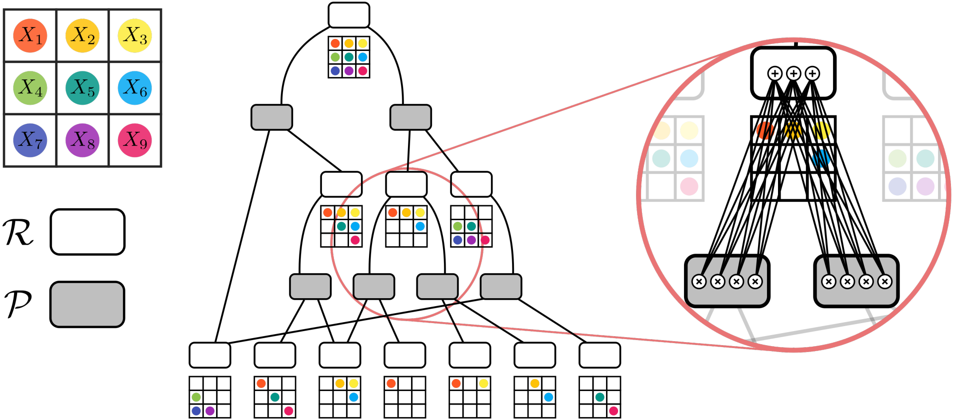
From region graphs to PCs

Equip partitions with products, combining units in input regions in all possible ways



From region graphs to PCs

Connect products to sum units above



From region graphs to PCs

- Equip each *input region* (leaf) $\mathcal{R}$ with K units $\phi_1, \ldots, \phi_K$, which are non-linear functions over $sc(\mathcal{R})$. Usually, $\phi_1, \ldots, \phi_K$ are probability densities. K can be different for each input region.
- Equip *each other region* with K sum units. K can be different for each internal region. Often, for the root region $K = 1$, when PC is used as density estimator.
- Equip each *partition* $\mathcal{P}$ with as many products as there are combinations of units in the input regions to $\mathcal{P}$, selecting one unit from each region. Formally, if $\mathcal{P}$ has input regions $\mathcal{R}_1, \mathcal{R}_2 \ldots, \mathcal{R}_I$, insert one product $\prod_{i=1}^{I} u_i$ for each $(u_1, u_2, \ldots, u_I) \in \mathcal{R}_1 \times \mathcal{R}_2 \times \cdots \times \mathcal{R}_I$.
- Connect each $\prod_{i=1}^{I} u_i$ in $\mathcal{P}$ to all sum units in the output regions of $\mathcal{P}$.

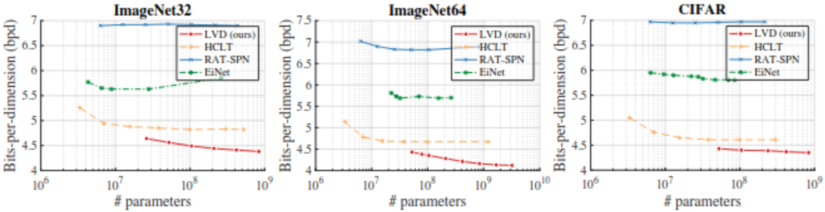
From region graphs to PCs

- Resulting PC has alternating sum and product units (not a strong constraint)
- We can easily scale the PC (overparameterize, increase expressivity) by equipping regions with more units
- RGs can be seen as a *vectorized version of PCs* – each region and partition can be seen as as a module
- Resulting PC will be *smooth and decomposable*, i.e., we can integrate, marginalize, and take conditionals
- After the PC has been constructed, we might discard the RG

Scaling up image models

Latent Variable Distillation

Dataset	TPMs				DGMs		
	LVD (ours)	HCLT	EiNet	RAT-SPN	Glow	RealNVP	BIVA
ImageNet32	**4.38**	4.82	5.63	6.90	4.09	4.28	3.96
ImageNet64	**4.12**	4.67	5.69	6.82	3.81	3.98	-
CIFAR	**4.37**	4.61	5.81	6.95	3.35	3.49	3.08



Liu et al., "Scaling Up Probabilistic Circuits by Latent Variable Distillation", 2022

How to construct and learn RGs?

Random regions graphs

The "no-learning" option *(Peharz et al. 2019)*

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

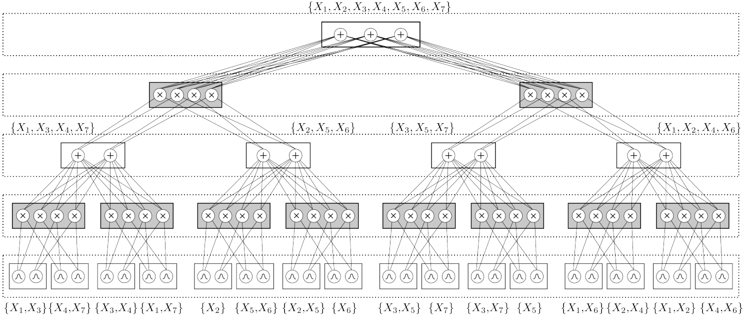
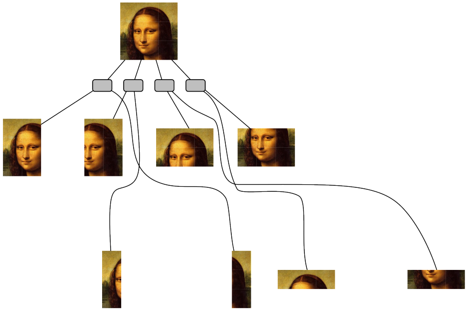


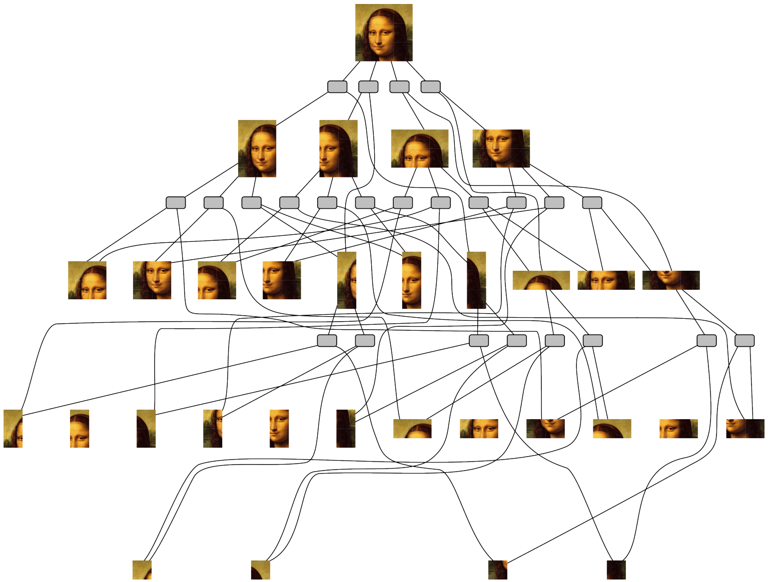
Image-tailored circuit structure

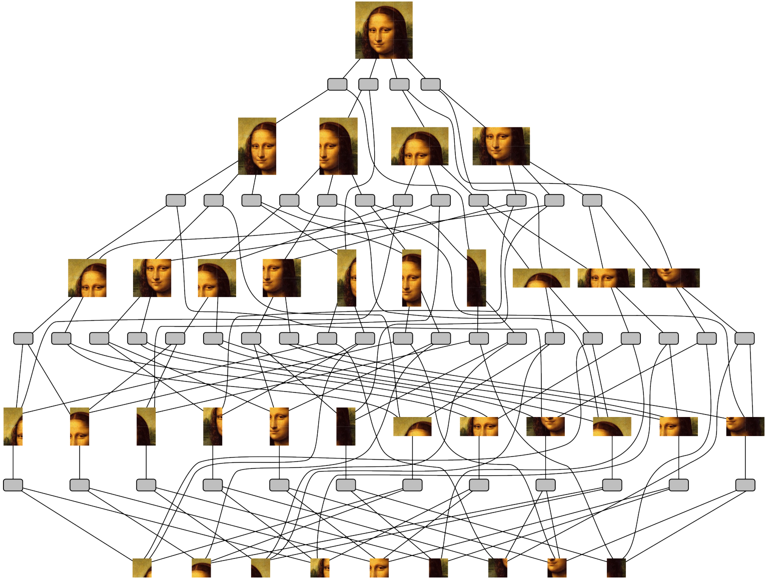
"Recursive image slicing" *(Poon et al. 2011)*

Images yield a natural region graph by using *axis-aligned splits*:

- Start with the full image (=output region)
- Define partitions by applying *horizontal* and *vertical* splits
- Recurse on the newly generated sub-images (internal regions)
- Structure somewhat reminiscent to convolutions
- Generates RGs which are "true DAGs," i.e. regions get re-used





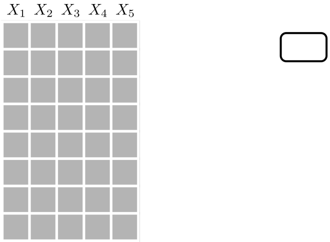


Data-driven structure learning

"Recursive data slicing"

(Gens et al. 2013)

Expand regions with **clustering**

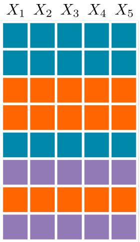


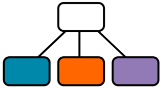
Data-driven structure learning

"Recursive data slicing"

(Gens et al. 2013)

Number of clusters = number of partitions

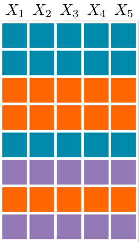


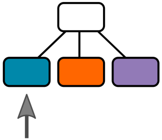


Data-driven structure learning

"Recursive data slicing" (Gens et al. 2013)

Try to find independent groups of variables
(e.g. independence tests)

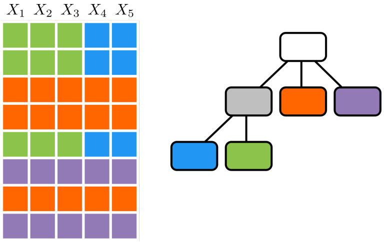




Data-driven structure learning

"Recursive data slicing" (Gens et al. 2013)

Success $\rightarrow$ ***partition*** into new regions

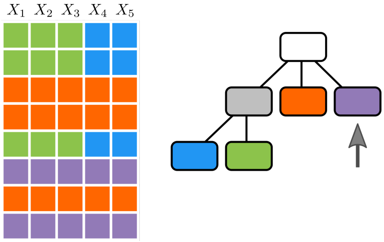


Data-driven structure learning

"Recursive data slicing"

(Gens et al. 2013)

Try to find independent groups of variables
(e.g. independence tests)

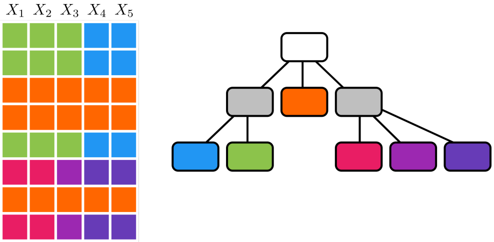


Data-driven structure learning

"Recursive data slicing"

(Gens et al. 2013)

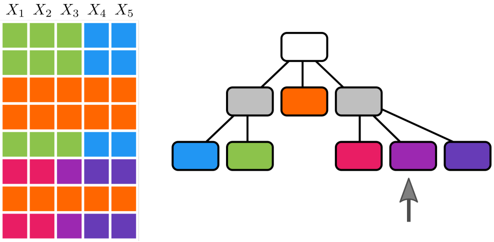
Success $\rightarrow$ ***partition*** into new regions



Data-driven structure learning

"Recursive data slicing" (Gens et al. 2013)

Single variable

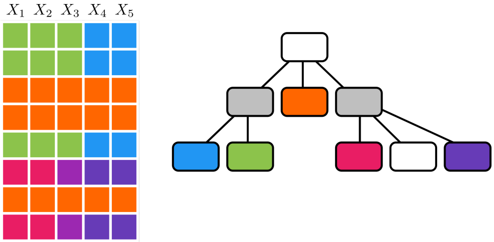


Data-driven structure learning

"Recursive data slicing"

(Gens et al. 2013)

Single variable $\longrightarrow$ ***input region***

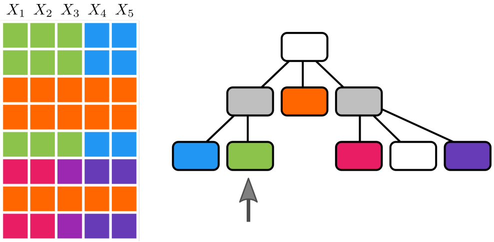


Data-driven structure learning

“Recursive data slicing”

(Gens et al. 2013)

Expand regions with **clustering**



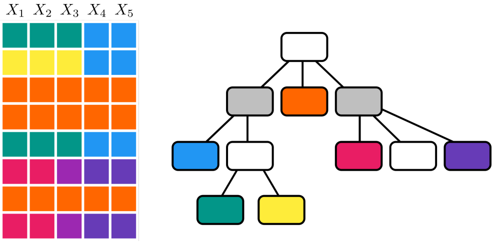
Data-driven structure learning

"Recursive data slicing"

(Gens et al. 2013)

Number of clusters = number of partitions

And so on...

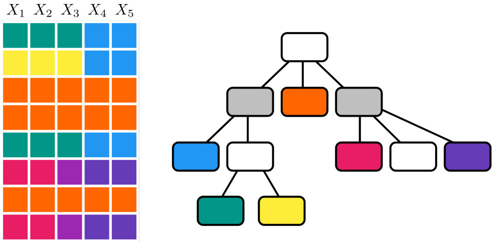


Data-driven structure learning

"Recursive data slicing"

(Gens et al. 2013)

- Stopping conditions: minimal number of features, samples, depth, ...
- Clustering ratios also deliver (initial) parameters
- **Smooth & Decomposable Circuits**
- **Tractable integration**



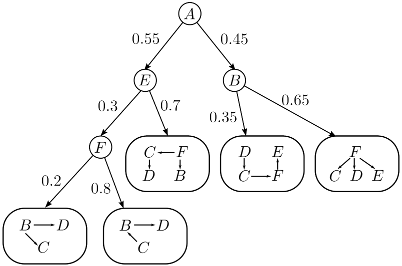
LearnSPN

Selected references

- **ID-SPN** *(Rooshenas et al. 2014)*
- **LearnSPN-b/T/B** *(Vergari et al. 2015)*
- For **heterogeneous data** *(Molina et al. 2018)*
- Using **k-means** *(Butz et al. 2018)* or **SVD** splits *(Adel et al. 2015)*
- Learning **DAGs** *(Dennis et al. 2015; Jaini et al. 2018)*
- **Approximating** independence tests *(Di Mauro et al. 2018)*

Cutset networks

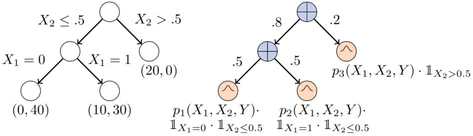
Besides clustering, **decision tree learning** can be used as PC learner. **Cutset networks,** decision trees over simple probabilistic models (Chow-Liu trees) *(Rahman et al. 2014):*

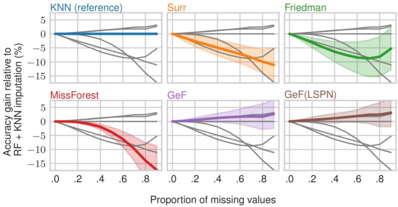


Cutset networks can easily be converted into **smooth, decomposable and deterministic PCs**.

Decision trees as PCs

Also vanilla decision tree learners can be used to learn PCs, by augmenting the leaves with distributions over inputs *(Correia et al. 2020).* Allows to treat **missing features** and **outlier detection**.





Information

Prior Knowledge
domain assumptions
constraints
other models

Data
experimental data
samples
measurements

compilation

learning

Circuits

Structure

Parameters

decomposability
smoothness
determinism
compatibility

$\boldsymbol{\theta}, \boldsymbol{w}$

generative
discriminative
Bayesian
credal

References I

- Chow, C and C Liu (1968). "Approximating discrete probability distributions with dependence trees". In: IEEE Transactions on Information Theory 14.3, pp. 462–467.
- Valiant, Leslie G (1979). "The complexity of enumeration and reliability problems". In: SIAM Journal on Computing 8.3, pp. 410–421.
- Bryant, R (1986). "Graph-based algorithms for boolean manipulation". In: IEEE Transactions on Computers, pp. 677–691.
- Cooper, Gregory F (1990). "The computational complexity of probabilistic inference using Bayesian belief networks". In: Artificial intelligence 42.2-3, pp. 393–405.
- Dagum, Paul and Michael Luby (1993). "Approximating probabilistic inference in Bayesian belief networks is NP-hard". In: Artificial intelligence 60.1, pp. 141–153.
- Zhang, Nevin Lianwen and David Poole (1994). "A simple approach to Bayesian network computations". In: Proceedings of the Biennial Conference-Canadian Society for Computational Studies of Intelligence, pp. 171–178.
- Roth, Dan (1996). "On the hardness of approximate reasoning". In: Artificial Intelligence 82.1–2, pp. 273–302.
- Dechter, Rina (1998). "Bucket elimination: A unifying framework for probabilistic inference". In: Learning in graphical models. Springer, pp. 75–104.
- Dasgupta, Sanjoy (1999). "Learning polytrees". In: Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., pp. 134–141.
- Hoey, Jesse, Robert St-Aubin, Alan Hu, and Craig Boutilier (1999). "SPUDD: stochastic planning using decision diagrams". In: Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, pp. 279–288.
- Meilă, Marina and Michael I. Jordan (2000). "Learning with mixtures of trees". In: Journal of Machine Learning Research 1, pp. 1–48.

References II

- Bach, Francis R. and Michael I. Jordan (2001). "Thin Junction Trees". In: Advances in Neural Information Processing Systems 14. MIT Press, pp. 569–576.
- Darwiche, Adnan (2001). "Recursive conditioning". In: Artificial Intelligence 126.1-2, pp. 5–41.
- Yedidia, Jonathan S, William T Freeman, and Yair Weiss (2001). "Generalized belief propagation". In: Advances in neural information processing systems, pp. 689–695.
- Darwiche, Adnan (2002). "A logical approach to factoring belief networks". In: KR 2, pp. 409–420.
- Darwiche, Adnan and Pierre Marquis (2002a). "A knowledge compilation map". In: Journal of Artificial Intelligence Research 17, pp. 229–264.
- — (2002b). "A knowledge compilation map". In: Journal of Artificial Intelligence Research 17.1, pp. 229–264.
- Dechter, Rina, Kalev Kask, and Robert Mateescu (2002). "Iterative join-graph propagation". In: Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., pp. 128–136.
- Darwiche, Adnan (2003). "A Differential Approach to Inference in Bayesian Networks". In: J.ACM.
- — (2004). "New advances in compiling CNF to decomposable negation normal form". In: Proc. of ECAI. Citeseer, pp. 328–332.
- Jaeger, Manfred (2004). "Probabilistic decision graphs—combining verification and AI techniques for probabilistic inference". In: International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 12.supp01, pp. 19–42.
- Sang, Tian, Paul Beame, and Henry A Kautz (2005). "Performing Bayesian inference by weighted model counting". In: AAAI. Vol. 5, pp. 475–481.

References III

- Sanner, Scott and David McAllester (2005). “Affine algebraic decision diagrams (AADDs) and their application to structured probabilistic inference”. In: IJCAI. Vol. 2005, pp. 1384–1390.
- Chavira, Mark, Adnan Darwiche, and Manfred Jaeger (2006). “Compiling relational Bayesian networks for exact inference”. In: International Journal of Approximate Reasoning 42.1-2, pp. 4–20.
- Jaeger, Manfred, Jens D Nielsen, and Tomi Silander (2006). “Learning probabilistic decision graphs”. In: International Journal of Approximate Reasoning 42.1-2, pp. 84–100.
- Park, James D and Adnan Darwiche (2006). “Complexity results and approximation strategies for MAP explanations”. In: Journal of Artificial Intelligence Research 21, pp. 101–133.
- De Raedt, Luc, Angelika Kimmig, and Hannu Toivonen (2007). “ProbLog: A Probabilistic Prolog and Its Application in Link Discovery.”. In: IJCAI. Vol. 7. Hyderabad, pp. 2462–2467.
- Dechter, Rina and Robert Mateescu (2007). “AND/OR search spaces for graphical models”. In: Artificial intelligence 171.2-3, pp. 73–106.
- Marinescu, Radu and Rina Dechter (2007). “Best-first AND/OR search for 0/1 integer programming”. In: International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming. Springer, pp. 171–185.
- Riguzzi, Fabrizio (2007). “A top down interpreter for LPAD and CP-logic”. In: Congress of the Italian Association for Artificial Intelligence. Springer, pp. 109–120.
- Chavira, Mark and Adnan Darwiche (2008). “On probabilistic inference by weighted model counting”. In: Artificial Intelligence 172.6-7, pp. 772–799.

References IV

- Olteanu, Dan and Jiewen Huang (2008). “Using OBDDs for efficient query evaluation on probabilistic databases”. In: International Conference on Scalable Uncertainty Management. Springer, pp. 326–340.
- Darwiche, Adnan (2009). Modeling and Reasoning with Bayesian Networks. Cambridge.
- Koller, Daphne and Nir Friedman (2009). Probabilistic Graphical Models: Principles and Techniques. MIT Press.
- Choi, Arthur and Adnan Darwiche (2010). “Relax, compensate and then recover”. In: JSAI International Symposium on Artificial Intelligence. Springer, pp. 167–180.
- Darwiche, Adnan (2011a). “SDD: A New Canonical Representation of Propositional Knowledge Bases”. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two. IJCAI’11. Barcelona, Catalonia, Spain. isbn: 978-1-57735-514-4.
- — (2011b). “SDD: A new canonical representation of propositional knowledge bases”. In: Twenty-Second International Joint Conference on Artificial Intelligence.
- de Campos, Cassio P (2011). “New complexity results for MAP in Bayesian networks”. In: IJCAI. Vol. 11, pp. 2100–2106.
- Poon, Hoifung and Pedro Domingos (2011). “Sum-Product Networks: a New Deep Architecture”. In: UAI 2011.
- Sontag, David, Amir Globerson, and Tommi Jaakkola (2011). “Introduction to dual decomposition for inference”. In: Optimization for Machine Learning 1, pp. 219–254.
- Muise, Christian, Sheila A McIlraith, J Christopher Beck, and Eric I Hsu (2012). “Dsharp: fast d-DNNF compilation with sharpSAT”. In: Canadian Conference on Artificial Intelligence. Springer, pp. 356–361.

References V

- Gens, Robert and Pedro Domingos (2013). “Learning the Structure of Sum-Product Networks”. In: Proceedings of the ICML 2013, pp. 873–880.
- Lowd, Daniel and Amirmohammad Rooshenas (2013). “Learning Markov Networks With Arithmetic Circuits”. In: Proceedings of the 16th International Conference on Artificial Intelligence and Statistics. Vol. 31. JMLR Workshop Proceedings, pp. 406–414.
- Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). “Generative adversarial nets”. In: Advances in neural information processing systems, pp. 2672–2680.
- Kingma, Diederik P and Max Welling (2014). “Auto-Encoding Variational Bayes”. In: Proceedings of the 2nd International Conference on Learning Representations (ICLR). 2014.
- Kisa, Doga, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche (July 2014). “Probabilistic sentential decision diagrams”. In: Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR). Vienna, Austria.
- Martens, James and Venkatesh Medabalimi (2014). “On the Expressive Efficiency of Sum Product Networks”. In: CoRR abs/1411.7717.
- Peharz, Robert, Robert Gens, and Pedro Domingos (2014). “Learning Selective Sum-Product Networks”. In: Workshop on Learning Tractable Probabilistic Models. LTPM.
- Rahman, Tahrima, Prasanna Kothalkar, and Vibhav Gogate (2014). “Cutset Networks: A Simple, Tractable, and Scalable Approach for Improving the Accuracy of Chow-Liu Trees”. In: Machine Learning and Knowledge Discovery in Databases. Vol. 8725. LNCS. Springer, pp. 630–645.
- Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). “Stochastic backprop. and approximate inference in deep generative models”. In: arXiv preprint arXiv:1401.4082.

References VI

- Rooshenas, Amirmohammad and Daniel Lowd (2014). "Learning Sum-Product Networks with Direct and Indirect Variable Interactions". In: Proceedings of ICML 2014.
- Adel, Tameem, David Balduzzi, and Ali Ghodsi (2015). "Learning the Structure of Sum-Product Networks via an SVD-based Algorithm". In: Uncertainty in Artificial Intelligence.
- Bekker, Jessa, Jesse Davis, Arthur Choi, Adnan Darwiche, and Guy Van den Broeck (2015). "Tractable Learning for Complex Probability Queries". In: Advances in Neural Information Processing Systems 28 (NIPS).
- Bova, Simone, Florent Capelli, Stefan Mengel, and Friedrich Slivovsky (2015). "On compiling CNFs into structured deterministic DNNFs". In: International Conference on Theory and Applications of Satisfiability Testing. Springer, pp. 199–214.
- Choi, Arthur, Guy Van den Broeck, and Adnan Darwiche (2015a). "Tractable learning for structured probability spaces: A case study in learning preference distributions". In: Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI).
- Choi, Arthur, Guy Van Den Broeck, and Adnan Darwiche (2015b). "Tractable Learning for Structured Probability Spaces: A Case Study in Learning Preference Distributions". In: Proceedings of the 24th International Conference on Artificial Intelligence. IJCAI'15. Buenos Aires, Argentina: AAAI Press, pp. 2861–2868. isbn: 978-1-57735-738-4. url: `http://dl.acm.org/citation.cfm?id=2832581.2832649`.
- Dennis, Aaron and Dan Ventura (2015). "Greedy Structure Search for Sum-product Networks". In: IJCAI'15. Buenos Aires, Argentina: AAAI Press, pp. 932–938. isbn: 978-1-57735-738-4.
- Di Mauro, Nicola, Antonio Vergari, and Floriana Esposito (2015). "Learning Accurate Cutset Networks by Exploiting Decomposability". In: Proceedings of AIXIA. Springer, pp. 221–232.

References VII

- Fierens, Daan, Guy Van den Broeck, Joris Renkens, Dimitar Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt (May 2015). "Inference and Learning in Probabilistic Logic Programs using Weighted Boolean Formulas". In: Theory and Practice of Logic Programming 15 (03), pp. 358–401. issn: 1475-3081. doi: 10.1017/S1471068414000076. url: http://starai.cs.ucla.edu/papers/FierensTPLP15.pdf.
- Germain, Mathieu, Karol Gregor, Iain Murray, and Hugo Larochelle (2015). "MADE: Masked Autoencoder for Distribution Estimation". In: CoRR abs/1502.03509.
- Peharz, Robert, Sebastian Tschiatschek, Franz Pernkopf, and Pedro Domingos (2015). "On Theoretical Properties of Sum-Product Networks". In: The Journal of Machine Learning Research.
- Vergari, Antonio, Nicola Di Mauro, and Floriana Esposito (2015). "Simplifying, Regularizing and Strengthening Sum-Product Network Structure Learning". In: ECML-PKDD 2015.
- Vlasselaer, Jonas, Guy Van den Broeck, Angelika Kimmig, Wannes Meert, and Luc De Raedt (2015). "Anytime Inference in Probabilistic Logic Programs with Tp-compilation". In: Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI). url: http://starai.cs.ucla.edu/papers/VlasselaerIJCAI15.pdf.
- Belle, Vaishak and Luc De Raedt (2016). "Semiring Programming: A Framework for Search, Inference and Learning". In: arXiv preprint arXiv:1609.06954.
- Bova, Simone (2016). "SDDs are exponentially more succinct than OBDDs". In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 30. 1.
- Bova, Simone, Florent Capelli, Stefan Mengel, and Friedrich Slivovsky (2016). "Knowledge Compilation Meets Communication Complexity.". In: IJCAI. Vol. 16, pp. 1008–1014.
- Cohen, Nadav, Or Sharir, and Amnon Shashua (2016). "On the expressive power of deep learning: A tensor analysis". In: Conference on Learning Theory, pp. 698–728.

References VIII

- Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2016). “Density estimation using real nvp”. In: arXiv preprint arXiv:1605.08803.
- Friesen, Abram L and Pedro Domingos (2016). “Submodular Sum-product Networks for Scene Understanding”. In.
- Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu (2016). “Pixel recurrent neural networks”. In: arXiv preprint arXiv:1601.06759.
- Oztok, Umut, Arthur Choi, and Adnan Darwiche (2016). “Solving PP-PP-complete problems using knowledge compilation”. In: Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning.
- Peharz, Robert, Robert Gens, Franz Pernkopf, and Pedro M. Domingos (2016). “On the Latent Variable Interpretation in Sum-Product Networks”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence PP, Issue 99. url: `http://arxiv.org/abs/1601.06180`.
- Sguerra, Bruno Massoni and Fabio G Cozman (2016). “Image classification using sum-product networks for autonomous flight of micro aerial vehicles”. In: 2016 5th Brazilian Conference on Intelligent Systems (BRACIS). IEEE, pp. 139–144.
- Shen, Yujia, Arthur Choi, and Adnan Darwiche (2016). “Tractable Operations for Arithmetic Circuits of Probabilistic Models”. In: Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, pp. 3936–3944.
- Vlasselaer, Jonas, Wannes Meert, Guy Van den Broeck, and Luc De Raedt (Mar. 2016). “Exploiting Local and Repeated Structure in Dynamic Bayesian Networks”. In: Artificial Intelligence 232, pp. 43 –53. issn: 0004-3702. doi: `10.1016/j.artint.2015.12.001`.
- Yuan, Zehuan, Hao Wang, Limin Wang, Tong Lu, Shivakumara Palaiahnakote, and Chew Lim Tan (2016). “Modeling spatial layout for scene image understanding via a novel multiscale sum-product network”. In: Expert Systems with Applications 63, pp. 231–240.

References IX

- Choi, YooJung, Adnan Darwiche, and Guy Van den Broeck (2017). "Optimal feature selection for decision robustness in Bayesian networks". In: Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI).
- Conaty, Diarmaid, Denis Deratani Mauá, and Cassio Polpo de Campos (2017). "Approximation Complexity of Maximum A Posteriori Inference in Sum-Product Networks". In: Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence. Ed. by Gal Elidan and Kristian Kersting. AUAI Press, pp. 322–331.
- Kimmig, Angelika, Guy Van den Broeck, and Luc De Raedt (2017). "Algebraic model counting". In: Journal of Applied Logic 22, pp. 46–62.
- Lagniez, Jean-Marie and Pierre Marquis (2017). "An Improved Decision-DNNF Compiler.". In: IJCAI. Vol. 17, pp. 667–673.
- Liang, Yitao and Guy Van den Broeck (Aug. 2017). "Towards Compact Interpretable Models: Shrinking of Learned Probabilistic Sentential Decision Diagrams". In: IJCAI 2017 Workshop on Explainable Artificial Intelligence (XAI). url: http://starai.cs.ucla.edu/papers/LiangXAI17.pdf.
- Papamakarios, George, Theo Pavlakou, and Iain Murray (2017). "Masked autoregressive flow for density estimation". In: Advances in Neural Information Processing Systems, pp. 2338–2347.
- Rathke, Fabian, Mattia Desana, and Christoph Schnörr (2017). "Locally adaptive probabilistic models for global segmentation of pathological oct scans". In: International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, pp. 177–184.
- Van den Broeck, Guy and Dan Suciu (Aug. 2017). Query Processing on Probabilistic Data: A Survey. Foundations and Trends in Databases. Now Publishers. doi: 10.1561/1900000052. url: http://starai.cs.ucla.edu/papers/VdBFTDB17.pdf.
- Butz, Cory J, Jhonatan S Oliveira, André E Santos, André L Teixeira, Pascal Poupart, and Agastya Kalra (2018). "An Empirical Study of Methods for SPN Learning and Inference". In: International Conference on Probabilistic Graphical Models, pp. 49–60.

References X

- Choi, YooJung and Guy Van den Broeck (2018). "On robust trimming of Bayesian network classifiers". In: arXiv preprint arXiv:1805.11243.
- Di Mauro, Nicola, Floriana Esposito, Fabrizio Giuseppe Ventola, and Antonio Vergari (2018). "Sum-Product Network structure learning by efficient product nodes discovery". In: Intelligenza Artificiale 12.2, pp. 143–159.
- Jaini, Priyank, Amur Ghose, and Pascal Poupart (2018). "Prometheus: Directly Learning Acyclic Directed Graph Structures for Sum-Product Networks". In: International Conference on Probabilistic Graphical Models, pp. 181–192.
- Molina, Alejandro, Antonio Vergari, Nicola Di Mauro, Sriraam Natarajan, Floriana Esposito, and Kristian Kersting (2018). "Mixed Sum-Product Networks: A Deep Architecture for Hybrid Domains". In: AAAI.
- Oztok, Umut and Adnan Darwiche (2018). "An exhaustive DPLL algorithm for model counting". In: Journal of Artificial Intelligence Research 62, pp. 1–32.
- Shen, Yujia, Arthur Choi, and Adnan Darwiche (2018). "Conditional PSDDs: Modeling and learning with modular knowledge". In: Thirty-Second AAAI Conference on Artificial Intelligence.
- Holtzen, Steven, Todd Millstein, and Guy Van den Broeck (2019). "Symbolic Exact Inference for Discrete Probabilistic Programs". In: arXiv preprint arXiv:1904.02079.
- Khosravi, Pasha, YooJung Choi, Yitao Liang, Antonio Vergari, and Guy Van den Broeck (2019a). "On Tractable Computation of Expected Predictions". In: Advances in Neural Information Processing Systems, pp. 11167–11178.
- Khosravi, Pasha, Yitao Liang, YooJung Choi, and Guy Van den Broeck (2019b). "What to Expect of Classifiers? Reasoning about Logistic Regression with Missing Features". In: Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI).

References XI

- Kossen, Jannik, Karl Stelzner, Marcel Hussing, Claas Voelcker, and Kristian Kersting (2019). “Structured Object-Aware Physics Prediction for Video Modeling and Planning”. In: arXiv preprint arXiv:1910.02425.
- Molina, Alejandro, Antonio Vergari, Karl Stelzner, Robert Peharz, Pranav Subramani, Nicola Di Mauro, Pascal Poupart, and Kristian Kersting (2019). “SPFlow: An easy and extensible library for deep probabilistic learning using sum-product networks”. In: arXiv preprint arXiv:1901.03704.
- Peharz, Robert, Antonio Vergari, Karl Stelzner, Alejandro Molina, Xiaoting Shao, Martin Trapp, Kristian Kersting, and Zoubin Ghahramani (2019). “Random sum-product networks: A simple but effective approach to probabilistic deep learning”. In: Proceedings of UAI.
- Shao, Xiaoting, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, and Kristian Kersting (2019). “Conditional Sum-Product Networks: Imposing Structure on Deep Probabilistic Architectures”. In: arXiv preprint arXiv:1905.08550.
- Shih, Andy, Guy Van den Broeck, Paul Beame, and Antoine Amarilli (2019). “Smoothing Structured Decomposable Circuits”. In: arXiv preprint arXiv:1906.00311.
- Stelzner, Karl, Robert Peharz, and Kristian Kersting (2019). “Faster Attend-Infer-Repeat with Tractable Probabilistic Models”. In: Proceedings of the 36th International Conference on Machine Learning. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, pp. 5966–5975. url: `http://proceedings.mlr.press/v97/stelzner19a.html`.
- Correia, Alvaro, Robert Peharz, and Cassio P de Campos (2020). “Joints in random forests”. In: Advances in neural information processing systems 33, pp. 11404–11415.
- Giunchiglia, Eleonora and Thomas Lukasiewicz (2020). “Coherent hierarchical multi-label classification networks”. In: NeurIPS 33, pp. 9662–9673.
- Peharz, Robert, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van den Broeck, Kristian Kersting, and Zoubin Ghahramani (2020). “Einsum Networks: Fast and Scalable Learning of Tractable Probabilistic Circuits”. In: ICML.

References XII

- Vlastelica, Marin, Anselm Paulus, Vit Musil, Georg Martius, and Michal Rolinek (2020). “Differentiation of blackbox combinatorial solvers”. In: ICLR.
- Colnet, Alexis de and Stefan Mengel (2021). “A Compilation of Succinctness Results for Arithmetic Circuits”. In: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning. Vol. 18. 1, pp. 205–215.
- **Vergari**, Antonio, YooJung Choi, Anji Liu, Stefano Teso, and Guy Van den Broeck (2021). “A Compositional Atlas of Tractable Circuit Operations: From Simple Transformations to Complex Information-Theoretic Queries”. In: NeurIPS. arXiv: `2102.06137 [stat.ML]`.
- Ahmed, Kareem, Stefano Teso, Kai-Wei Chang, Guy Van den Broeck, and Antonio Vergari (2022a). “Semantic Probabilistic Layers for Neuro-Symbolic Learning”. In: arXiv preprint arXiv:2206.00426.
- Ahmed, Kareem, Eric Wang, Kai-Wei Chang, and Guy Van den Broeck (2022b). “Neuro-symbolic entropy regularization”. In: Uncertainty in Artificial Intelligence. PMLR, pp. 43–53.
- Dang, Meihua, Anji Liu, and Guy Van den Broeck (2022). “Sparse Probabilistic Circuits via Pruning and Growing”. In: NeurIPS. url: `http://starai.cs.ucla.edu/papers/DangNeurIPS22.pdf`.
- Liu, Anji, Honghua Zhang, and Guy Van den Broeck (2022). “Scaling Up Probabilistic Circuits by Latent Variable Distillation”. In: arXiv preprint.