Probabilistic Circuits

Representations
Inference
Learning
Applications

Antonio Vergari
University of California, Los Angeles

based on joint AAAI-2020 and UAI-2019 tutorials with

Guy Van den Broeck
University of California, Los Angeles

YooJung Choi
University of California, Los Angeles

Robert Peharz
TU Eindhoven

Nicola Di Mauro
University of Bari

December 2nd, 2019 - ***"Deep Generative Models"*** **- Stanford, CA**

Fully factorized

NaiveBayes *AndOrGraphs* *PDGs*

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

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

Polytrees *FVSBNs* *TACs* *IAFs* *NAFs* *RAEs*

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

RealNVP *MNs*

The Alphabet Soup of probabilistic models

Fully factorized

NaiveBayes AndOrGraphs PDGs

Trees PSDDs CNets LTMs SPNs NADEs

Thin Junction Trees ACs MADEs MAFs VAEs

Polytrees FVSBNs TACs IAFs NAFs RAEs

Mixtures BNs NICE FGs GANs

RealNVP MNs

Intractable and *tractable* models

Fully factorized

NaiveBayes *AndOrGraphs* *PDGs*

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

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

Polytrees *FVSBNs* *TACs* *IAFs* *NAFs* *RAEs*

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

RealNVP *MNs*

tractability is a spectrum

Fully factorized

NaiveBayes AndOrGraphs PDGs

Trees PSDDs CNets LTMs SPNs NADEs

Thin Junction Trees ACs MADEs MAFs VAEs

Polytrees FVSBNs TACs IAFs NAFs RAEs

Mixtures BNs NICE FGs GANs

RealNVP MNs

Expressive models without *compromises*

Fully factorized

NaiveBayes AndOrGraphs PDGs

Trees PSDDs CNets LTMs SPNs NADEs

Thin Junction Trees ACs MADEs MAFs VAEs

Polytrees FVSBNs TACs IAFs NAFs RAEs

Mixtures BNs NICE FGs GANs

RealNVP MNs

a *unifying framework* for tractable models

Why tractable inference?

or expressiveness vs tractability

Why tractable inference?

or expressiveness vs tractability

Probabilistic circuits

a unified framework for tractable models

Why tractable inference?

or expressiveness vs tractability

Probabilistic circuits

a unified framework for tractable models

Building circuits

learning them from data and compiling other models

Why tractable inference?

or expressiveness vs tractability

Probabilistic circuits

a unified framework for tractable models

Building circuits

learning them from data and compiling other models

Applications

what are circuits useful for

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 Alma Str.?*

Why probabilistic inference?

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

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

© fineartamerica.com

Why probabilistic inference?

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

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

$\Longrightarrow$ fitting a predictive model!

© fineartamerica.com

Why probabilistic inference?

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

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

$\Longrightarrow$ ~~fitting a predictive model!~~

Why probabilistic inference?

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

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

$\implies$ ~~fitting a predictive model!~~

$\implies$ answering probabilistic ***queries*** on a probabilistic model of the world m

$$q_1(m) = ? \qquad q_2(m) = ?$$

Why probabilistic inference?

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

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

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

Why probabilistic inference?

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

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

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

$\Longrightarrow$ ***marginals***

Why probabilistic inference?

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

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

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

© fineartamerica.com

Why probabilistic inference?

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

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

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

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

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

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

Tractable Probabilistic Inference

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

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

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

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****!*

Why exact inference?

or "What about approximate inference?"

1. No need for approximations when we can be exact
2. We can do exact inference in approximate models *[Dechter et al. 2002; Choi et al. 2010; Lowd et al. 2010; Sontag et al. 2011; Friedman et al. 2018]*
3. Approximations shall come with guarantees
4. Approximate inference (even with guarantees) can mislead learners *[Kulesza et al. 2007]*
5. Approximations can be intractable as well *[Dagum et al. 1993; Roth 1996]*

Why exact inference?

or "What about approximate inference?"

1. No need for approximations when we can be exact

   ⇒ *do we lose some expressiveness?*

2. We can do exact inference in approximate models *[Dechter et al. 2002; Choi et al. 2010; Lowd et al. 2010; Sontag et al. 2011; Friedman et al. 2018]*
3. Approximations shall come with guarantees
4. Approximate inference (even with guarantees) can mislead learners *[Kulesza et al. 2007]*
5. Approximations can be intractable as well *[Dagum et al. 1993; Roth 1996]*

Why exact inference?

or "What about approximate inference?"

1. No need for approximations when we can be exact
2. We can do exact inference in approximate models *[Dechter et al. 2002; Choi et al. 2010; Lowd et al. 2010; Sontag et al. 2011; Friedman et al. 2018]*
3. Approximations shall come with guarantees

   $\Rightarrow$ *sometimes they do, e.g., [Dechter et al. 2007]*
4. Approximate inference (even with guarantees) can mislead learners *[Kulesza et al. 2007]*
5. Approximations can be intractable as well *[Dagum et al. 1993; Roth 1996]*

Why exact inference?

or "What about approximate inference?"

1. No need for approximations when we can be exact
2. We can do exact inference in approximate models *[Dechter et al. 2002; Choi et al. 2010; Lowd et al. 2010; Sontag et al. 2011; Friedman et al. 2018]*
3. Approximations shall come with guarantees
4. Approximate inference (even with guarantees) can mislead learners *[Kulesza et al. 2007]* $\Longrightarrow$ *Chaining approximations is flying with a blindfold on*
5. Approximations can be intractable as well *[Dagum et al. 1993; Roth 1996]*

Why exact inference?

or "What about approximate inference?"

1. No need for approximations when we can be exact
2. We can do exact inference in approximate models *[Dechter et al. 2002; Choi et al. 2010; Lowd et al. 2010; Sontag et al. 2011; Friedman et al. 2018]*
3. Approximations shall come with guarantees
4. Approximate inference (even with guarantees) can mislead learners *[Kulesza et al. 2007]*
5. Approximations can be intractable as well *[Dagum et al. 1993; Roth 1996]*

Stay tuned for...

Next:

1. *What are classes of queries?*
2. *Are my favorite models tractable?*
3. *Are tractable models expressive?*

After:

We introduce **probabilistic circuits** *as a unified framework for tractable probabilistic modeling*

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 Alma Str.?*

© fineartamerica.com

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 Alma Str.?*

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

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

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 Alma Str.?*

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

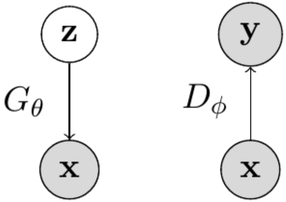
$$\mathbf{q}_3(\mathbf{m}) = p_{\mathbf{m}}(\mathbf{X} = \{\mathsf{Mon}, 12.00, 1, 0, \ldots, 0\})$$

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

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

Generative Adversarial Networks

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

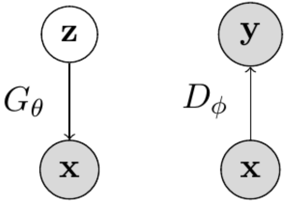


Goodfellow et al., "Generative adversarial nets", 2014

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*

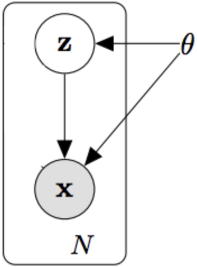


Goodfellow et al., "Generative adversarial nets", 2014

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 et al., "Auto-Encoding Variational Bayes", 2014

~~Variational Autoencoders~~

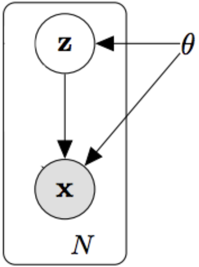
$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x} \mid \mathbf{z})\right] - \mathbb{KL}(q_\phi(\mathbf{z} \mid \mathbf{x})||p(\mathbf{z}))$$

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

  $\Rightarrow$ *an infinite and uncountable mixture*

  $\Rightarrow$ *no tractable EVI*
- we need to optimize the ELBO...

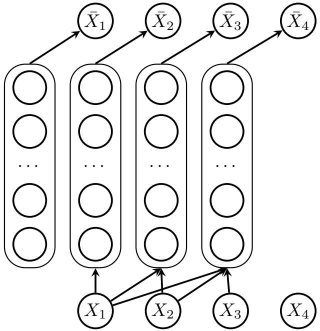
  $\Rightarrow$ *which is "tricky" [Alemi et al. 2017; Dai et al. 2019; Ghosh et al. 2019]*



Autoregressive models

$$p_\theta(\mathbf{x}) = \prod_i p_\theta(x_i \mid x_1, x_2, \ldots, x_{i-1})$$

- an explicit likelihood!
- ...as a product of factors $\Longrightarrow$ *tractable EVI!*
- many neural variants
  - NADE *[Larochelle et al. 2011]*, MADE *[Germain et al. 2015]*
  - PixelCNN *[Salimans et al. 2017]*, 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 Alma Str.?*

© 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 Alma Str.?*

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

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 Alma Str.?*

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

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

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

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 Alma Str.?*

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

General: $p_{\mathbf{m}}(\mathbf{e}) = \int p_{\mathbf{m}}(\mathbf{e}, \mathbf{H})\, d\mathbf{H}$
and if you can answer MAR queries,
then you can also do ***conditional queries*** (CON):

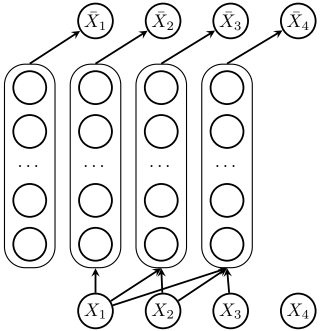
$$p_{\mathbf{m}}(\mathbf{q} \mid \mathbf{e}) = \frac{p_{\mathbf{m}}(\mathbf{q}, \mathbf{e})}{p_{\mathbf{m}}(\mathbf{e})}$$

© fineartamerica.com

Autoregressive models

$$p_\theta(\mathbf{x}) = \prod_i p_\theta(x_i \mid x_1, x_2, \ldots, x_{i-1})$$

- an explicit likelihood!
- ...as a product of factors $\Rightarrow$ *tractable EVI!*

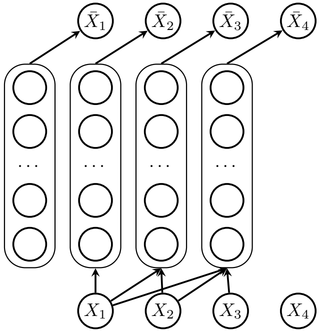


~~Autoregressive models~~

$$p_\theta(\mathbf{x}) = \prod_i p_\theta(x_i \mid x_1, x_2, \ldots, x_{i-1})$$

- an explicit likelihood!
- ...as a product of factors $\Rightarrow$ *tractable EVI!*
- ... but we need to fix a variable ordering

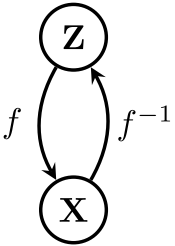
$\Rightarrow$ ***only some*** *MAR queries are tractable for one ordering*



Normalizing flows

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

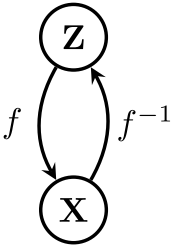
- an explicit likelihood $\Longrightarrow$ *tractable EVI!*
- ... computing the determinant of the Jacobian



~~Normalizing flows~~

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

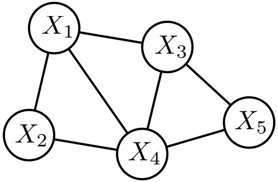
- an explicit likelihood $\implies$ *tractable EVI!*
- ... computing the determinant of the Jacobian
- MAR is generally intractable
  $\implies$ *unless f is a “trivial” bijection*



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-complete*

$\Longrightarrow$ *[Cooper 1990; Roth 1996]*

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

$\Longrightarrow$ *[Dagum et al. 1993; Roth 1996]*

Why? Treewidth!

Treewidth:

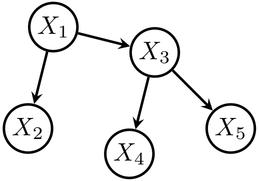
Informally, how tree-like is the graphical model $\mathbf{m}$?

Formally, the minimum width of any tree-decomposition of $\mathbf{m}$.

Fixed-parameter tractable: MAR and CON on a graphical model $\mathbf{m}$ with treewidth w take time $O(|\mathbf{X}| \cdot 2^w)$, which is linear for fixed width w

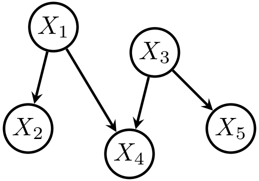
[Dechter 1998; Koller et al. 2009]. $\Longrightarrow$ *what about bounding the treewidth by design?*

Low-treewidth PGMs



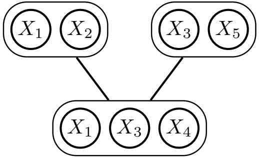
Trees

[Meilă et al. 2000]



Polytrees

[Dasgupta 1999]



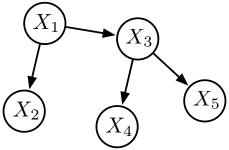
Thin Junction trees

[Bach et al. 2001]

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

What do we lose?

Expressiveness: Ability to represent rich and complex classes of distributions

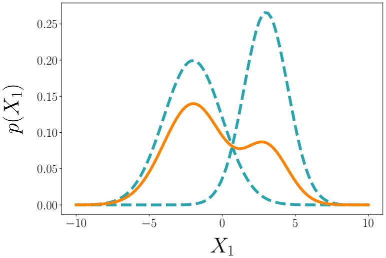


Bounded-treewidth PGMs lose the ability to represent *all possible distributions* ...

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

Mixtures

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

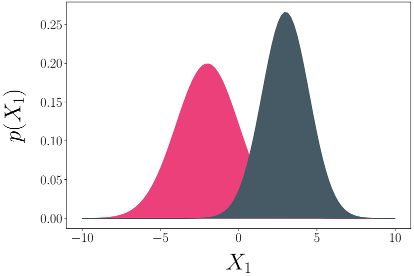


$$p(X) = w_1 \cdot p_1(X) + w_2 \cdot p_2(X)$$

EVI, MAR, CON queries scale linearly in k

Mixtures

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



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

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

$\Rightarrow$ *increased expressiveness*

Expressiveness and efficiency

Expressiveness: Ability to represent rich and effective classes of functions

$\Longrightarrow$ *mixture of Gaussians can approximate any distribution!*

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

Expressiveness and efficiency

Expressiveness: Ability to represent rich and effective classes of functions

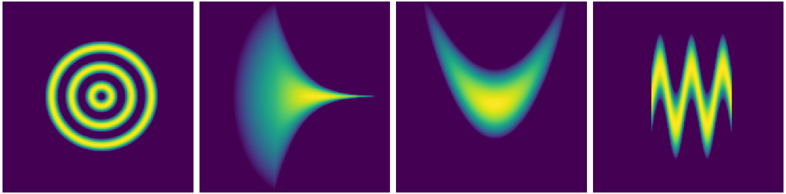
$\implies$ *mixture of Gaussians can approximate any distribution!*

Expressive efficiency (succinctness) Ability to represent rich and effective classes of functions **compactly**

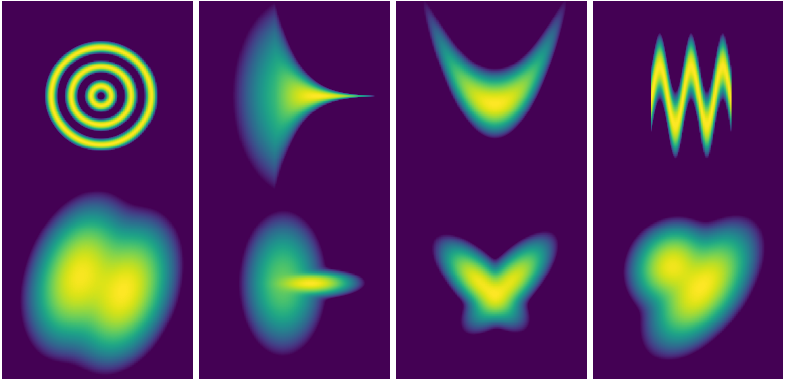
$\implies$ *but how many components does a Gaussian mixture need?*

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

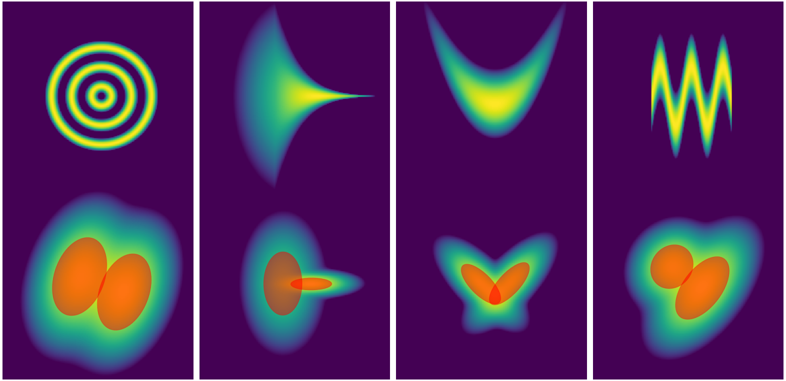
How expressive efficient are mixture?



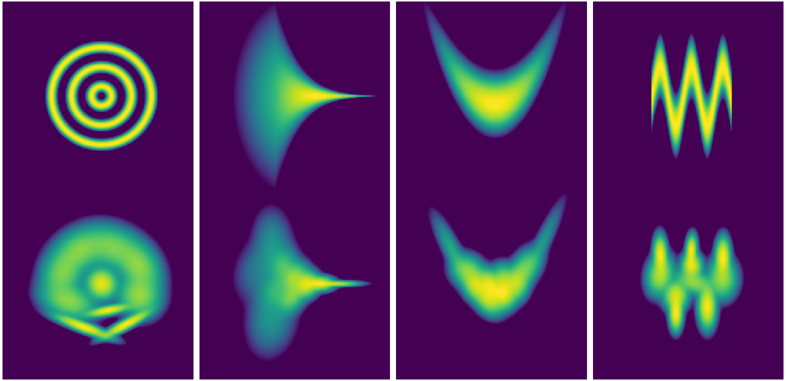
How expressive efficient are mixture?



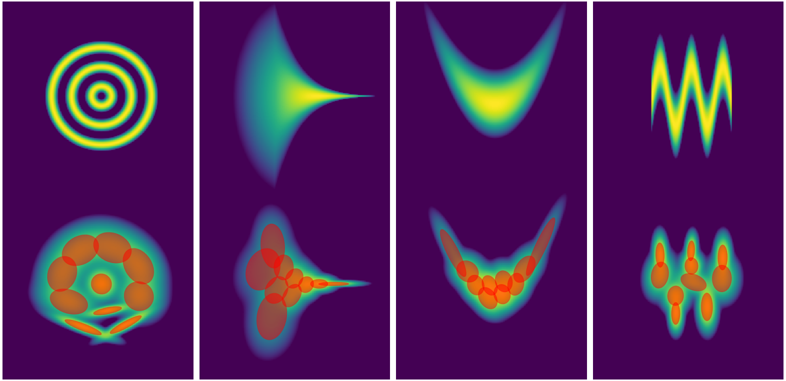
How expressive efficient are mixture?



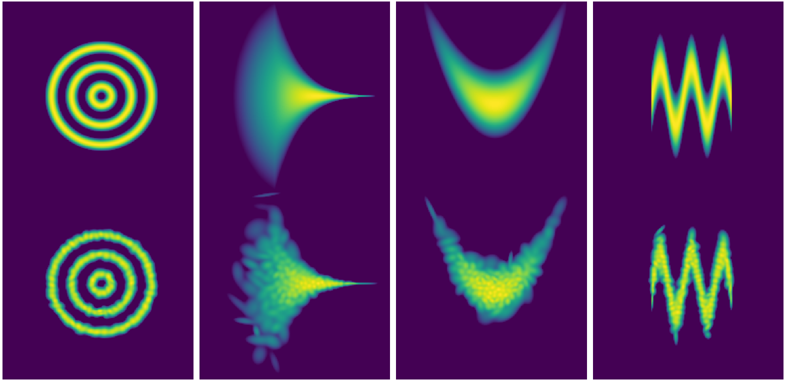
How expressive efficient are mixture?



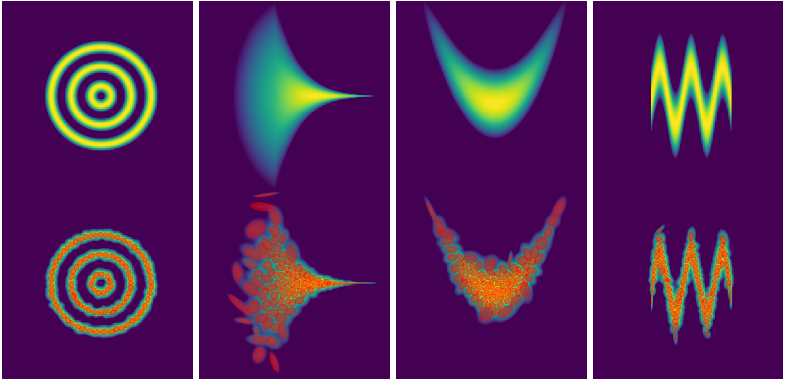
How expressive efficient are mixture?



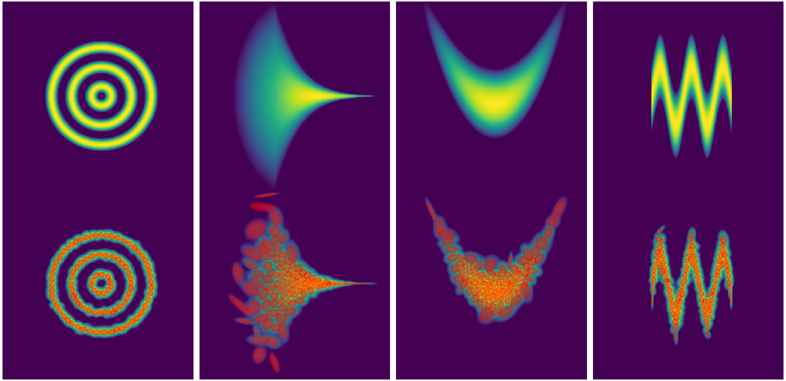
How expressive efficient are mixture?



How expressive efficient are mixture?



How expressive efficient are mixture?



⟹ *stack mixtures like in deep generative models*

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?*

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

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?*

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

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

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

***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

Marginal MAP (MMAP)

aka Bayesian Network MAP

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

Marginal MAP (MMAP)

aka Bayesian Network MAP

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

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

© 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?*

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

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

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

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?*

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

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

$\Longrightarrow$ *NP-hard for trees* [Campos 2011]

$\Longrightarrow$ *NP-hard even for Naive Bayes* [ibid.]

Advanced queries

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

© fineartamerica.com

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

Advanced queries

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

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

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

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 work?*

q_7: *What is the probability of seeing more traffic jams in Palo Verde than Midtown?*

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 work?*

q_7: *What is the probability of seeing more traffic jams in Palo Verde than Midtown?*

$\Rightarrow$ **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 work?*

q_7: *What is the probability of seeing more traffic jams in Palo Verde than Midtown?*

and more:

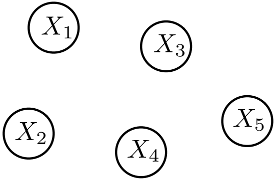
- expected classification agreement *[Oztok et al. 2016; Choi et al. 2017, 2018]*
- expected predictions *[Khosravi et al. 2019b]*

© fineartamerica.com

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

Fully factorized models

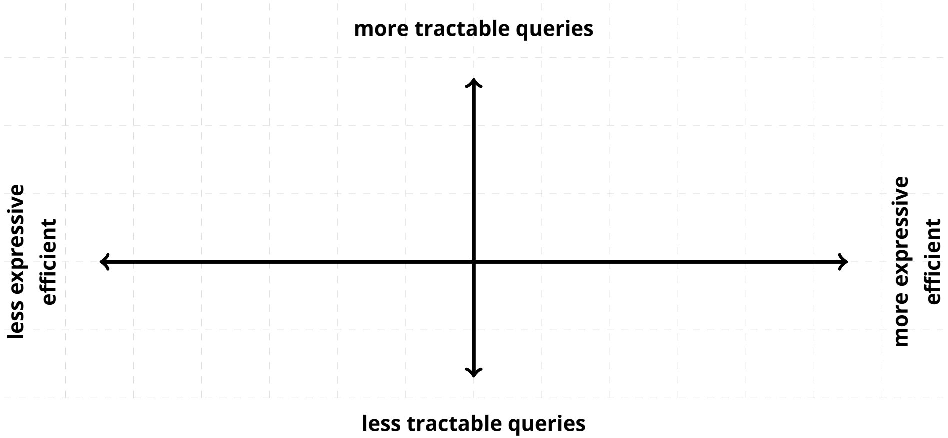
A completely disconnected graph. Example: Product of Bernoullis (PoBs)

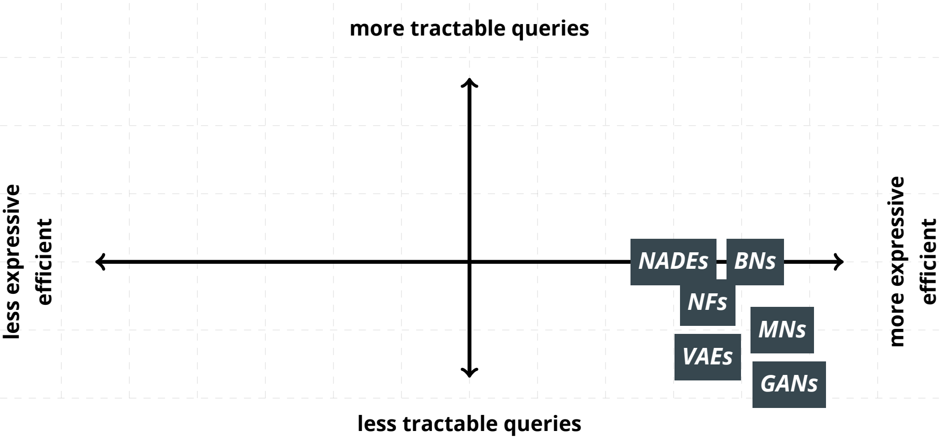


$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i)$$

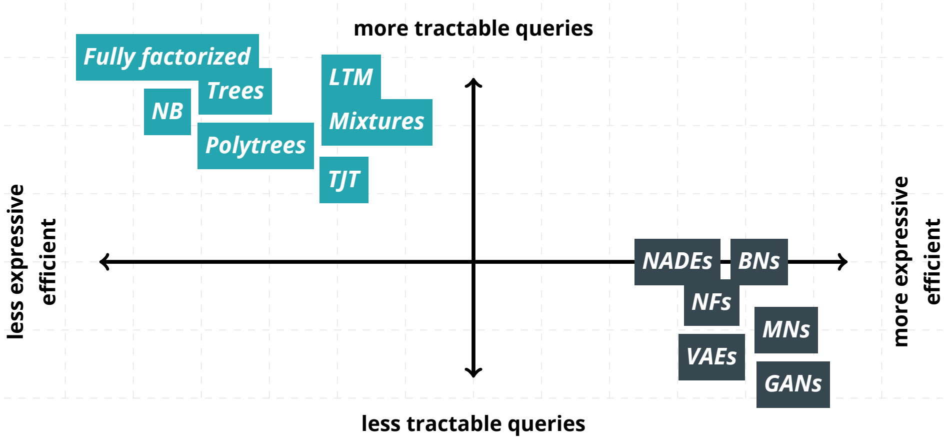
Complete evidence, marginals and MAP, MMAP inference is ***linear***!

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

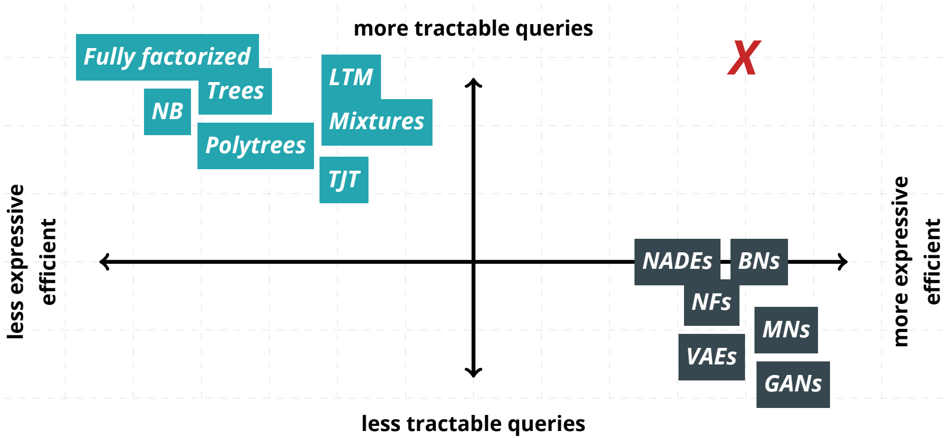

more tractable queries
less expressive
efficient
more expressive
efficient
less tractable queries



Expressive models are not very tractable...



and *tractable* ones are not very expressive...



probabilistic circuits are at the "sweet spot"

Probabilistic Circuits

Probabilistic circuits

A probabilistic circuit $\mathcal{C}$ *over variables* $\mathbf{X}$ *is a computational graph encoding a (possibly unnormalized) probability distribution* $p(\mathbf{X})$

Probabilistic circuits

A probabilistic circuit $\mathcal{C}$ *over variables* $\mathbf{X}$ *is a computational graph encoding a (possibly unnormalized) probability distribution* $p(\mathbf{X})$

$\Longrightarrow$ operational semantics!

Probabilistic circuits

A probabilistic circuit $\mathcal{C}$ *over variables* $\mathbf{X}$ *is a computational graph encoding a (possibly unnormalized) probability distribution* $p(\mathbf{X})$

$\Longrightarrow$ operational semantics!

$\Longrightarrow$ by constraining the graph we can make inference tractable...

Stay tuned for...

Next:

1. *What are the building blocks of probabilistic circuits?*

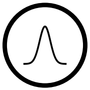
   ⇒ *How to build a tractable computational graph?*

2. For which queries are probabilistic circuits tractable?

   ⇒ *tractable classes induced by structural properties*

After: *How can probabilistic circuits be learned?*

Distributions as computational graphs

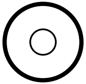


X

Base case: a single node encoding a distribution

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

Distributions as computational graphs

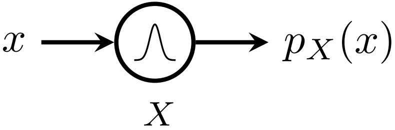


$\neg X$

Base case: a single node encoding a distribution

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

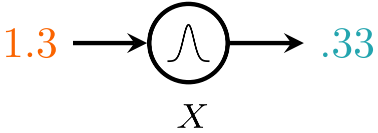
Distributions as computational graphs



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

Distributions as computational graphs



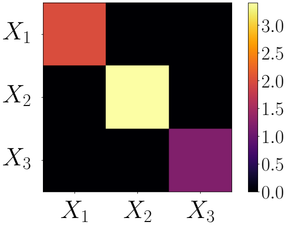
Simple distributions are tractable "black boxes" for:

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

Factorizations as product nodes

Divide and conquer complexity

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

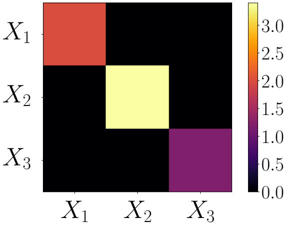


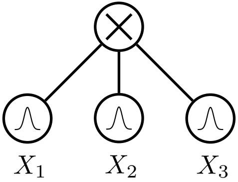
$\Rightarrow$ *e.g. modeling a multivariate Gaussian with diagonal covariance matrix...*

Factorizations as product nodes

Divide and conquer complexity

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



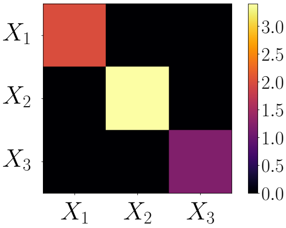


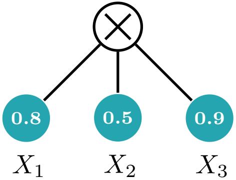
$\Rightarrow$ *...with a product node over some univariate Gaussian distribution*

Factorizations as product nodes

Divide and conquer complexity

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



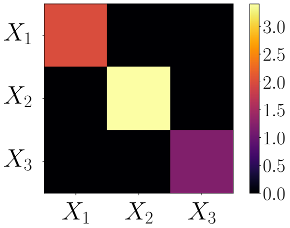


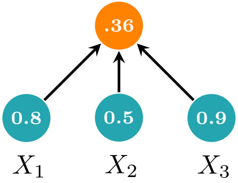
$\Rightarrow$ *feedforward evaluation*

Factorizations as product nodes

Divide and conquer complexity

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

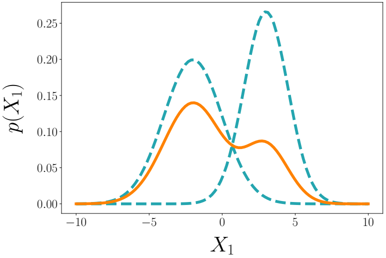




$\Longrightarrow$ *feedforward evaluation*

Mixtures as sum nodes

Enhance expressiveness

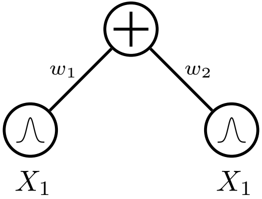


$$p(X) = w_1 \cdot p_1(X) + w_2 \cdot p_2(X)$$

$\Longrightarrow$ *e.g. modeling a mixture of Gaussians...*

Mixtures as sum nodes

Enhance expressiveness

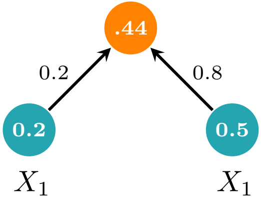


$$p(x) = 0.2 \cdot p_1(x) + 0.8 \cdot p_2(x)$$

$\Rightarrow$ *...as weighted a sum node over Gaussian input distributions*

Mixtures as sum nodes

Enhance expressiveness



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

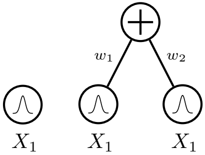
$\Longrightarrow$ *by **stacking** them we increase expressive efficiency*

A grammar for tractable models

Recursive semantics of probabilistic circuits

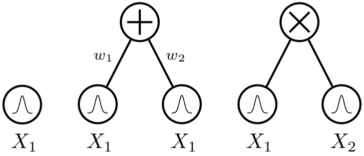
A grammar for tractable models

Recursive semantics of probabilistic circuits



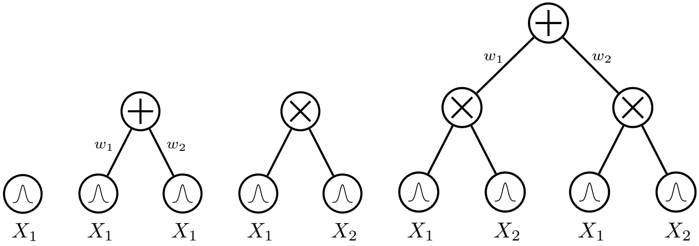
A grammar for tractable models

Recursive semantics of probabilistic circuits



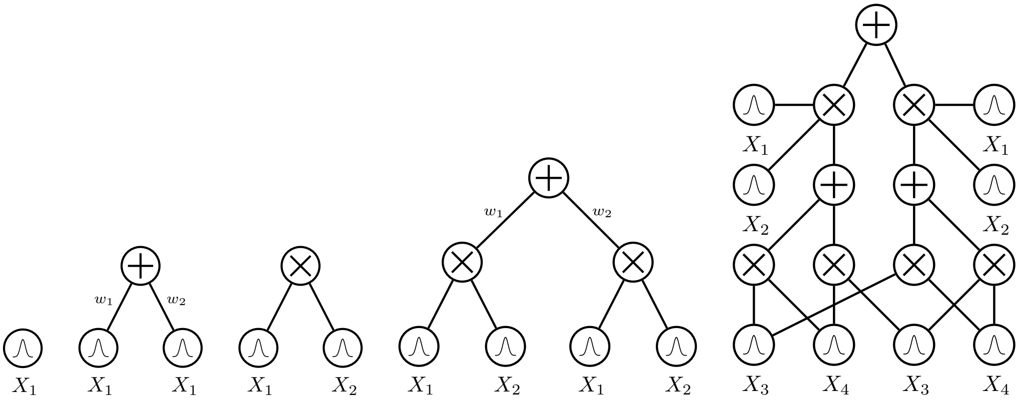
A grammar for tractable models

Recursive semantics of probabilistic circuits



A grammar for tractable models

Recursive semantics of probabilistic circuits



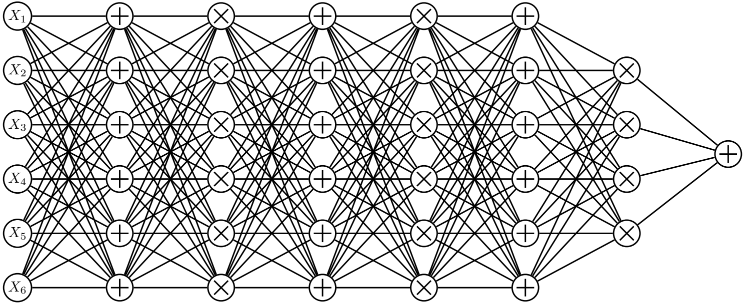
Probabilistic circuits are not PGMs!

They are ***probabilistic*** and ***graphical***, however ...

	PGMs	***Circuits***
Nodes:	random variables	unit of computations
Edges:	dependencies	order of execution
Inference:	conditioning elimination message passing	feedforward pass backward pass

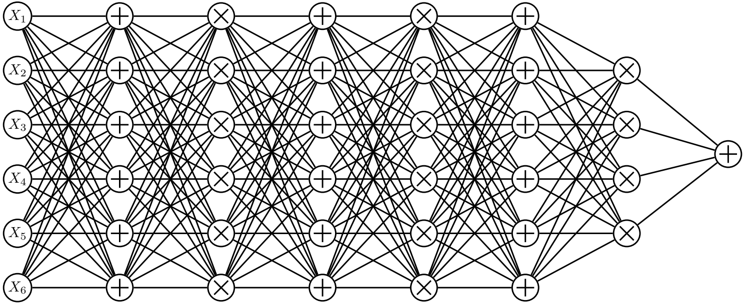
⇒ *they are **computational graphs**, more like neural networks*

Just sum, products and distributions?



just arbitrarily compose them like a neural network!

Just sum, products and distributions?



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

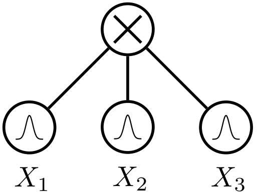
$\Rightarrow$ structural constraints needed for tractability

Which structural constraints to ensure tractability?

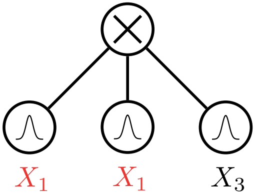
Decomposability

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

⇒ *just like in factorization!*



decomposable circuit



non-decomposable circuit

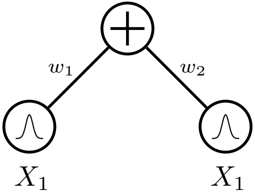
Darwiche et al., "A knowledge compilation map", 2002

Smoothness

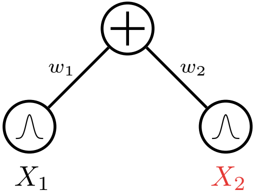
aka completeness

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

⟹ *otherwise not accounting for some variables*



smooth circuit



non-smooth circuit

⟹ *smoothness can be easily enforced [Shih et al. 2019]*

Darwiche et al., "A knowledge compilation map", 2002

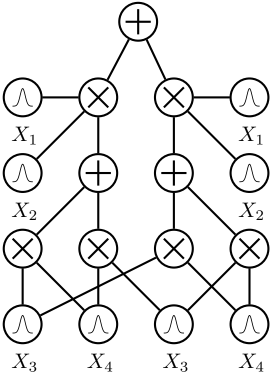
Smoothness + decomposability = tractable MAR

Computing arbitrary integrations (or summations)

$\Longrightarrow$ *linear in circuit size!*

E.g., suppose we want to compute Z:

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



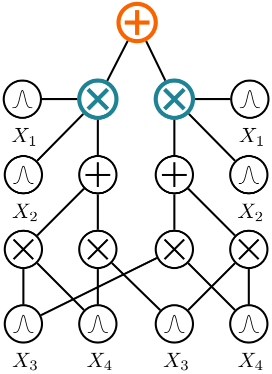
Smoothness + decomposability = tractable MAR

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

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

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

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



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

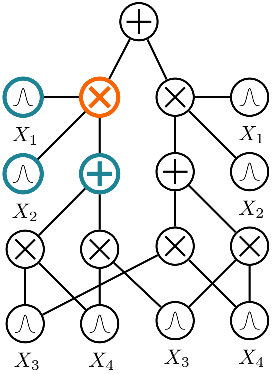
If $p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{x})p(\mathbf{y})p(\mathbf{z})$, (***decomposability***):

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

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

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

$\Rightarrow$ *larger integrals decompose into easier ones*



Smoothness + decomposability = tractable MAR

Forward pass evaluation for MAR

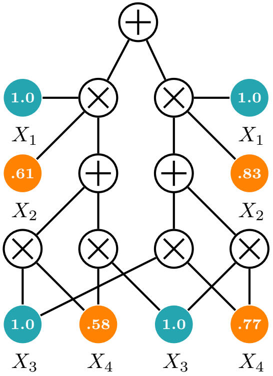
$\Longrightarrow$ *linear in circuit size!*

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

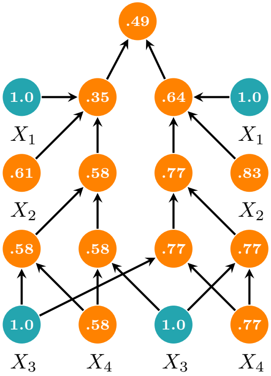
Forward pass evaluation for MAR

$\Rightarrow$ *linear in circuit size!*

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

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

  $\Rightarrow$ *for normalized leaf distributions:* 1.0
- leafs over X_2 and X_4 output ***EVI***
- feedforward evaluation (bottom-up)



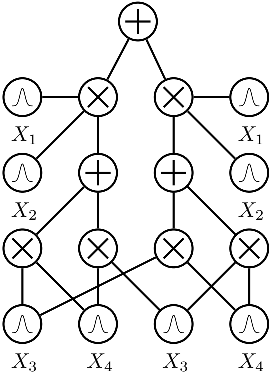
Smoothness + **decomposability** = **tractable CON**

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!*



Smoothness + *decomposability* = *tractable MAP*

We can also decompose bottom-up a MAP query:

$$\operatorname*{argmax}_{\mathbf{q}} p(\mathbf{q} \mid \mathbf{e})$$

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

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

$$\operatorname*{argmax}_{\mathbf{q}} p(\mathbf{q} \mid \mathbf{e})$$

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

$$\operatorname*{argmax}_{\mathbf{q}} \sum_i w_i p_i(\mathbf{q}, \mathbf{e}) = \operatorname*{argmax}_{\mathbf{q}} \sum_{\mathbf{z}} p(\mathbf{q}, \mathbf{z}, \mathbf{e}) \neq \sum_{\mathbf{z}} \operatorname*{argmax}_{\mathbf{q}} p(\mathbf{q}, \mathbf{z}, \mathbf{e})$$

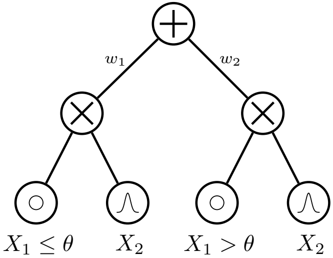
$\Longrightarrow$ *MAP for latent variable models is* **intractable** *[Conaty et al. 2017]*

Determinism

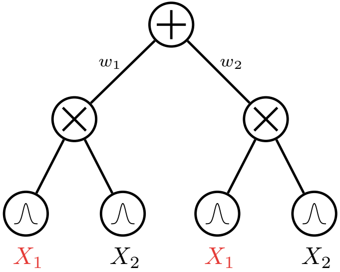
aka selectivity

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

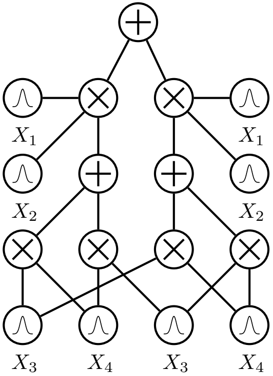
Determinism + decomposability = tractable MAP

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

$\Longrightarrow$ *linear in circuit size!*

E.g., suppose we want to compute:

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

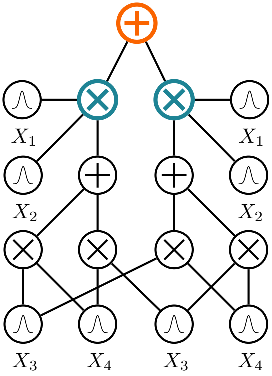


Determinism + decomposability = tractable MAP

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):

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

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

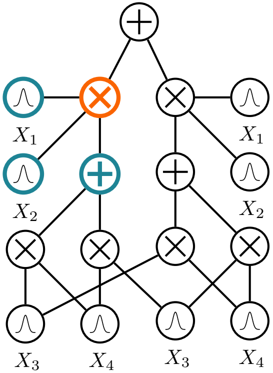


Determinism + **decomposability** = **tractable MAP**

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):

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

$\Longrightarrow$ *solving optimization independently*



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

Evaluating the circuit twice:

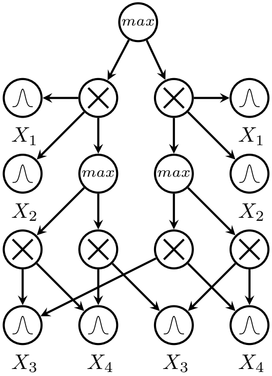
bottom-up and ***top-down*** $\Rightarrow$ *still linear in circuit size!*

Determinism + decomposability = tractable MAP

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

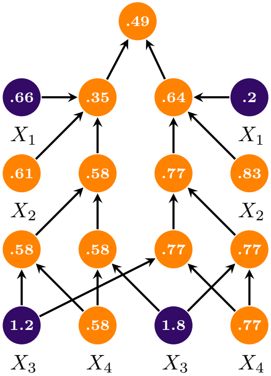


Determinism + ***decomposability*** = ***tractable MAP***

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

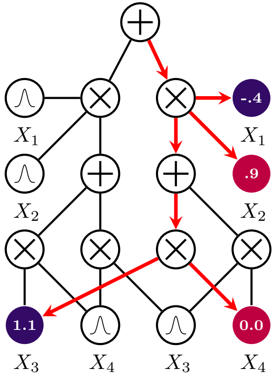


Determinism + decomposability = tractable MAP

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

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

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

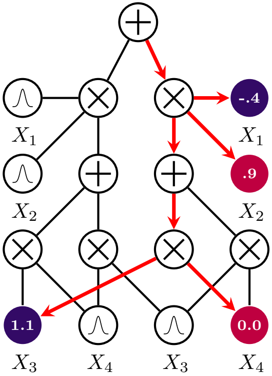


Determinism + decomposability = tractable MAP

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

Analogously, we could can also do a MMAP query:

$$\arg\max_{\mathbf{q}} \sum_{\mathbf{z}} p(\mathbf{q}, \mathbf{z} \mid \mathbf{e})$$

Determinism + **decomposability** = ~~**tractable MMAP**~~

We ***cannot*** decompose a MMAP query!

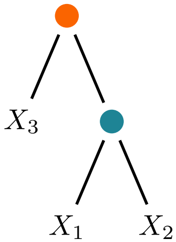
$$\operatorname*{argmax}_{\mathbf{q}} \sum_{\mathbf{z}} p(\mathbf{q}, \mathbf{z} \mid \mathbf{e})$$

we still have latent variables to marginalize…

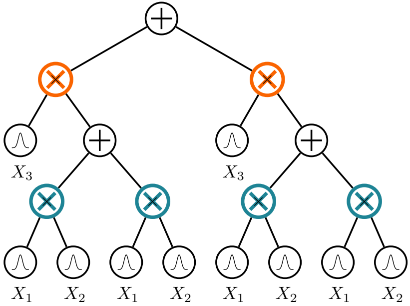
Structured decomposability

A product node is structured decomposable if decomposes according to a node in a ***vtree***

$\Rightarrow$ *stronger requirement than decomposability*



vtree

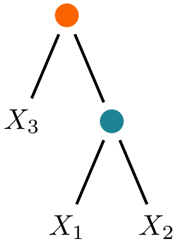


structured decomposable circuit

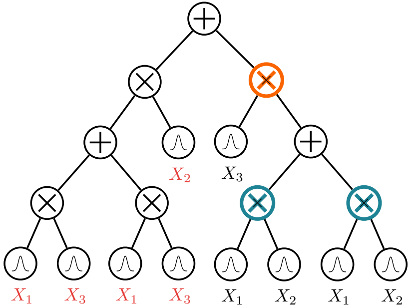
Structured decomposability

A product node is structured decomposable if decomposes according to a node in a ***vtree***

$\Rightarrow$ *stronger requirement than decomposability*



vtree



non structured decomposable circuit

structured decomposability = tractable...

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

For the "right" vtree

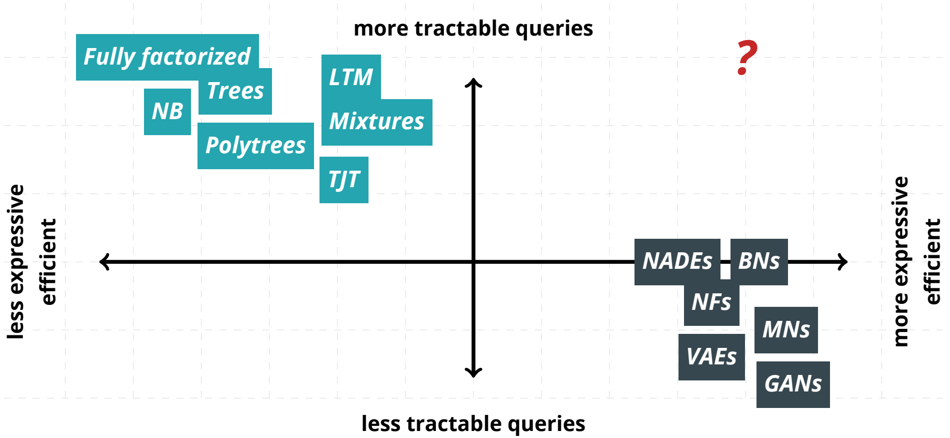
- Probability of logical circuit event in probabilistic circuit *[ibid.]*
- ***Multiply*** two probabilistic circuits *[Shen et al. 2016]*
- ***KL Divergence*** between probabilistic circuits *[Liang et al. 2017b]*
- ***Same-decision probability*** *[Oztok et al. 2016]*
- ***Expected same-decision probability***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

structured decomposability = ***tractable...***

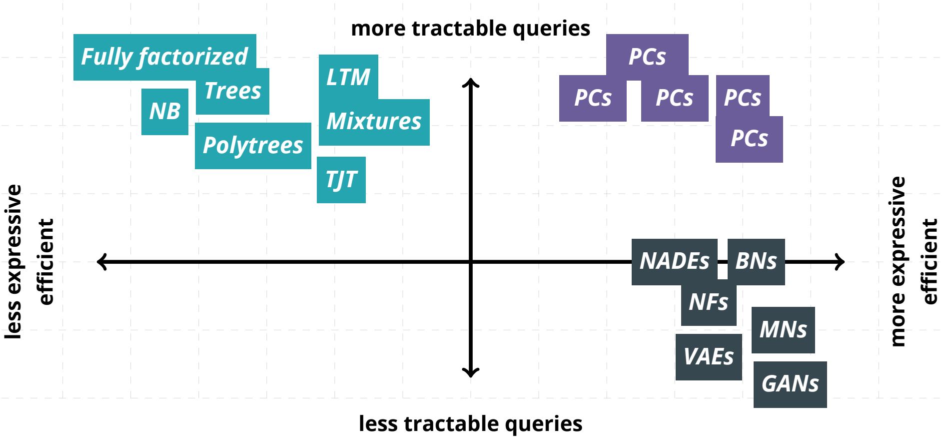
- ***Symmetric*** and ***group queries*** (exactly-k, odd-number, etc.) *[Bekker et al. 2015]*

For the "right" vtree

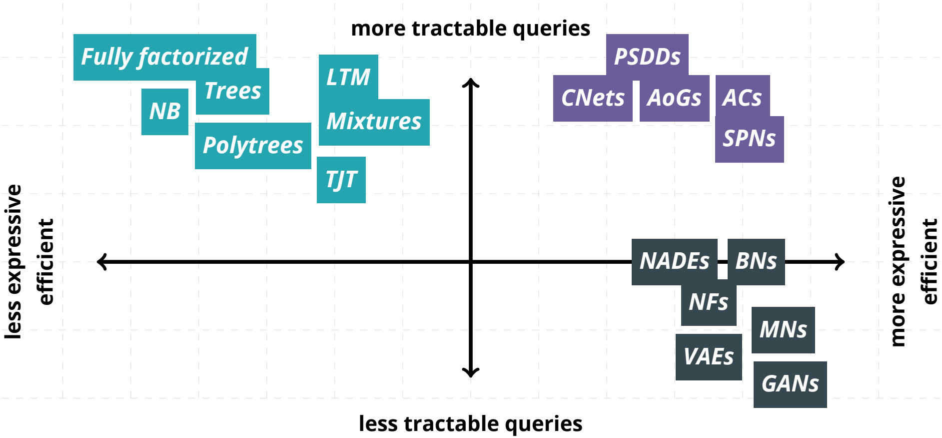
- Probability of logical circuit event in probabilistic circuit *[ibid.]*
- ***Multiply*** two probabilistic circuits *[Shen et al. 2016]*
- ***KL Divergence*** between probabilistic circuits *[Liang et al. 2017b]*
- ***Same-decision probability*** *[Oztok et al. 2016]*
- ***Expected same-decision probability***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



where are probabilistic circuits?



tractability vs expressive efficiency



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]*	✔	✔	✔	✘
PSDDs *[Kisa et al. 2014a]*	✔	✔	✔	✔
AndOrGraphs *[Dechter et al. 2007]*	✔	✔	✔	✔

How expressive are probabilistic circuits?

Measuring average test set log-likelihood on 20 density estimation benchmarks

Comparing against intractable models:

- Bayesian networks (BN) *[Chickering 2002]* with sophisticated context-specific CPDs
- MADEs *[Germain et al. 2015]*
- VAEs *[Kingma et al. 2014]* (IWAE ELBO *[Burda et al. 2015]*)

Gens et al., "Learning the Structure of Sum-Product Networks", 2013
peharz2018probabilistic, **peharz2018probabilistic**, **peharz2018probabilistic**

How expressive are probabilistic circuits?

density estimation benchmarks

dataset	best circuit	BN	MADE	VAE	dataset	best circuit	BN	MADE	VAE
nltcs	**-5.99**	-6.02	-6.04	**-5.99**	***dna***	**-79.88**	-80.65	-82.77	-94.56
msnbc	**-6.04**	**-6.04**	-6.06	-6.09	***kosarek***	**-10.52**	-10.83	-	-10.64
kdd	-2.12	-2.19	**-2.07**	-2.12	***msweb***	-9.62	-9.70	**-9.59**	-9.73
plants	**-11.84**	-12.65	-12.32	-12.34	***book***	-33.82	-36.41	-33.95	**-33.19**
audio	-39.39	-40.50	-38.95	**-38.67**	***movie***	-50.34	-54.37	-48.7	**-47.43**
jester	-51.29	**-51.07**	-52.23	-51.54	***webkb***	-149.20	-157.43	-149.59	**-146.9**
netflix	-55.71	-57.02	-55.16	**-54.73**	***cr52***	-81.87	-87.56	-82.80	**-81.33**
accidents	-26.89	**-26.32**	-26.42	-29.11	***c20ng***	-151.02	-158.95	-153.18	**-146.9**
retail	**-10.72**	-10.87	-10.81	-10.83	***bbc***	**-229.21**	-257.86	-242.40	-240.94
pumbs*	-22.15	**-21.72**	-22.3	-25.16	***ad***	-14.00	-18.35	**-13.65**	-18.81

Building circuits

Learning probabilistic circuits

A probabilistic circuit $\mathcal{C}$ *over variables* $\mathbf{X}$ *is a* **computational graph** *encoding a (possibly unnormalized) probability distribution* $p(\mathbf{X})$ *parameterized by* $\mathbf{\Omega}$

Learning probabilistic circuits

A probabilistic circuit $\mathcal{C}$ *over variables* $\mathbf{X}$ *is a* **computational graph** *encoding a (possibly unnormalized) probability distribution* $p(\mathbf{X})$ *parameterized by* $\mathbf{\Omega}$

Learning a circuit $\mathcal{C}$ *from data* $\mathcal{D}$ *can therefore involve learning the graph (****structure****) and/or its* ***parameters***

Learning probabilistic circuits

	Parameters	Structure
Generative	?	?
Discriminative	?	?

Stay tuned for...

Next:

1. *How to learn circuit parameters?*

   ⇒ *convex optimization, EM, SGD, Bayesian learning, ...*

2. *How to learn the structure of circuits?*

   ⇒ *local search, random structures, ensembles, ...*

After: *Which applications are circuits used for?*

Learning circuit parameters

Let a circuit structure $\mathcal{C}$ be given. We aim to learn its parameters:

- Parameters of input distributions
  $\boldsymbol{\theta} = \{\boldsymbol{\theta}_{\mathsf{L}}\}_{\mathsf{L} \in \mathsf{leaves}(\mathcal{C})}$

$\Longrightarrow$ *e.g.* $\boldsymbol{\theta}_{\mathsf{L}} = (\mu, \sigma)$ *if* L *is Gaussian, etc.*

Learning circuit parameters

Let a circuit structure $\mathcal{C}$ be given. We aim to learn its parameters:

- Parameters of input distributions
  $\boldsymbol{\theta} = \{\boldsymbol{\theta}_{\mathsf{L}}\}_{\mathsf{L} \in \mathsf{leaves}(\mathcal{C})}$
- Sum-weights $\mathbf{w} = \{\mathbf{w}_{\mathsf{S}}\}_{\mathsf{S} \in \mathsf{sums}(\mathcal{C})}$
  $\Longrightarrow$ *w.l.o.g., for each* S: $\sum_i w_{\mathsf{S},i} = 1$ *[Peharz et al. 2015; Zhao et al. 2015]*

Learning circuit parameters

Let a circuit structure $\mathcal{C}$ be given. We aim to learn its parameters:

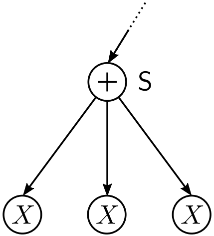
- Parameters of input distributions
  $\boldsymbol{\theta} = \{\boldsymbol{\theta}_{\mathsf{L}}\}_{\mathsf{L} \in \mathsf{leaves}(\mathcal{C})}$
- Sum-weights $\mathbf{w} = \{\mathbf{w}_{\mathsf{S}}\}_{\mathsf{S} \in \mathsf{sums}(\mathcal{C})}$

$\Longrightarrow$ *we marginalize out latent variable* Z_{S}

$$\mathcal{C}_{\mathsf{S}} = \sum_i \overbrace{p(Z_{\mathsf{S}} = i \mid \text{“}context\text{''})}^{w_{\mathsf{S},i}} \, \mathcal{C}_{\mathsf{N}_i}$$

Augmentation

Making latent variables explicit

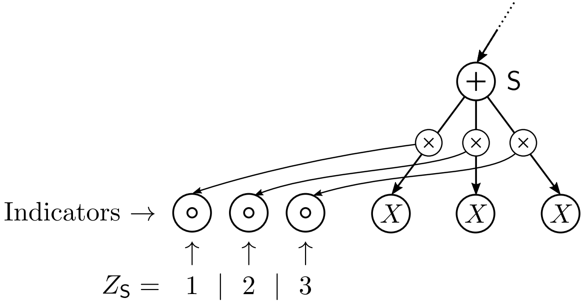


Peharz et al., "On the Latent Variable Interpretation in Sum-Product Networks", 2016

Augmentation

Making latent variables explicit

Setting single indicators to $1 \Rightarrow$ switches on corresponding child.

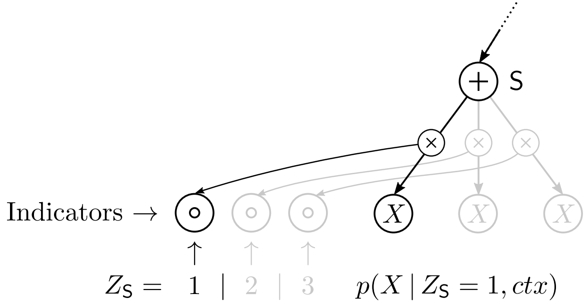


Peharz et al., "On the Latent Variable Interpretation in Sum-Product Networks", 2016

Augmentation

Making latent variables explicit

Yes, but we might have destroyed smoothness...

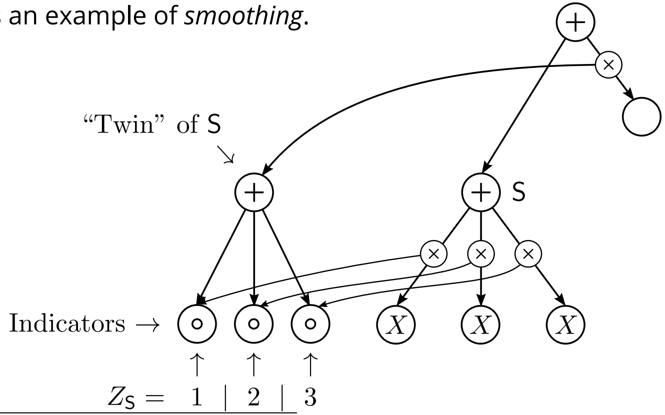


Peharz et al., "On the Latent Variable Interpretation in Sum-Product Networks", 2016

Augmentation

Making latent variables explicit

This is an example of *smoothing*.

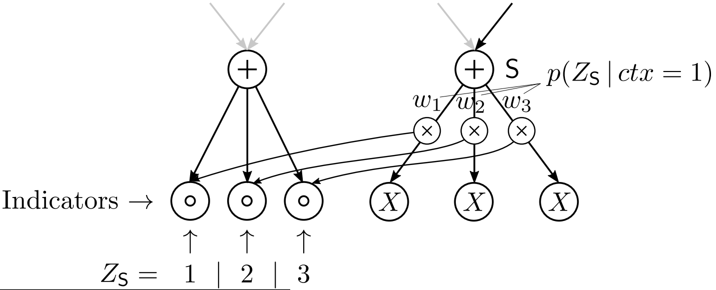


Peharz et al., "On the Latent Variable Interpretation in Sum-Product Networks", 2016

Augmentation

Making latent variables explicit

Thus, sum weights have sound probabilistic semantics.



Peharz et al., "On the Latent Variable Interpretation in Sum-Product Networks", 2016

Expectation-Maximization

Given a probabilistic circuit $\mathcal{C}$ and a dataset $\mathbf{D}$, the standard EM update is:

$$w_{i,j}^{new} = \frac{\sum_{\mathbf{x}\in\mathbf{D}} \mathbb{P}[ctx_i = 1 \land Z_i = j \,|\, \mathbf{x}, \mathbf{w}^{old}]}{\sum_{\mathbf{x}\in\mathbf{D}} \mathbb{P}[ctx_i = 1 \,|\, \mathbf{x}, \mathbf{w}^{old}]}$$

Peharz et al., "On the Latent Variable Interpretation in Sum-Product Networks", 2016

Expectation-Maximization

Given a probabilistic circuit $\mathcal{C}$ and a dataset $\mathbf{D}$, the standard EM update is:

$$w_{i,j}^{new} = \frac{\sum_{\mathbf{x}\in\mathbf{D}} \mathbb{P}[ctx_i = 1 \wedge Z_i = j \,|\, \mathbf{x}, \mathbf{w}^{old}]}{\sum_{\mathbf{x}\in\mathbf{D}} \mathbb{P}[ctx_i = 1 \,|\, \mathbf{x}, \mathbf{w}^{old}]}$$

These expected statistics can be computed efficiently with ***backprop*** *[Darwiche 2003]:*

$$\mathbb{P}[ctx_i = 1 \wedge Z_i = j \,|\, \mathbf{x}, \mathbf{w}^{old}] = \frac{1}{\mathcal{C}(\mathbf{x})} \frac{\partial \mathcal{C}(\mathbf{x})}{\partial \mathcal{C}_i(\mathbf{x})} \mathcal{C}_j(\mathbf{x}) w_{i,j}^{old}$$

Peharz et al., "On the Latent Variable Interpretation in Sum-Product Networks", 2016

Expectation-Maximization

Given a probabilistic circuit $\mathcal{C}$ and a dataset $\mathbf{D}$, the standard EM update is:

$$w_{i,j}^{new} = \frac{\sum_{\mathbf{x}\in\mathbf{D}} \mathbb{P}[ctx_i = 1 \wedge Z_i = j \,|\, \mathbf{x}, \mathbf{w}^{old}]}{\sum_{\mathbf{x}\in\mathbf{D}} \mathbb{P}[ctx_i = 1 \,|\, \mathbf{x}, \mathbf{w}^{old}]}$$

These expected statistics can be computed efficiently with ***backprop*** *[Darwiche 2003]:*

$$\mathbb{P}[ctx_i = 1 \wedge Z_i = j \,|\, \mathbf{x}, \mathbf{w}^{old}] = \frac{1}{\mathcal{C}(\mathbf{x})} \frac{\partial \mathcal{C}(\mathbf{x})}{\partial \mathcal{C}_i(\mathbf{x})} \mathcal{C}_j(\mathbf{x}) w_{i,j}^{old}$$

⟹ *This also works with missing values in $\mathbf{x}$!*

⟹ *Similar updates for leaves, when in exponential family.*

Peharz et al., "On the Latent Variable Interpretation in Sum-Product Networks", 2016

Deterministic Circuits

Exact Maximum Likelihood

Given a deterministic circuit $\mathcal{C}$ and a complete dataset $\mathbf{D}$, the maximum-likelihood sum-weights are:

$$w_{i,j}^{\mathsf{MLE}} = \frac{\sum_{\mathbf{x}\in\mathbf{D}} \mathbb{1}\{\mathbf{x} \models [i \wedge j]\}}{\sum_{\mathbf{x}\in\mathbf{D}} \mathbb{1}\{\mathbf{x} \models [i]\}}$$

Kisa et al., "Probabilistic sentential decision diagrams", 2014
Peharz et al., "Learning Selective Sum-Product Networks", 2014
Liang et al., "Learning Logistic Circuits", 2019

Deterministic Circuits

Exact Maximum Likelihood

Given a deterministic circuit $\mathcal{C}$ and a complete dataset $\mathbf{D}$, the maximum-likelihood sum-weights are:

$$w_{i,j}^{\text{MLE}} = \frac{\sum_{\mathbf{x}\in\mathbf{D}} \mathbb{1}\{\mathbf{x} \models [i \wedge j]\}}{\sum_{\mathbf{x}\in\mathbf{D}} \mathbb{1}\{\mathbf{x} \models [i]\}}$$

samples activating node j

Kisa et al., "Probabilistic sentential decision diagrams", 2014
Peharz et al., "Learning Selective Sum-Product Networks", 2014
Liang et al., "Learning Logistic Circuits", 2019

Deterministic Circuits

Exact Maximum Likelihood

Given a deterministic circuit $\mathcal{C}$ and a complete dataset $\mathbf{D}$, the maximum-likelihood sum-weights are:

$$w_{i,j}^{\mathsf{MLE}} = \frac{\sum_{\mathbf{x}\in\mathbf{D}} \mathbb{1}\{\mathbf{x} \models [i \wedge j]\}}{\sum_{\mathbf{x}\in\mathbf{D}} \mathbb{1}\{\mathbf{x} \models [i]\}}$$

samples activating node j

samples activating node i

Kisa et al., "Probabilistic sentential decision diagrams", 2014
Peharz et al., "Learning Selective Sum-Product Networks", 2014
Liang et al., "Learning Logistic Circuits", 2019

Deterministic Circuits

Exact Maximum Likelihood

Given a deterministic circuit $\mathcal{C}$ and a complete dataset $\mathbf{D}$, the maximum-likelihood sum-weights are:

$$w_{i,j}^{\mathsf{MLE}} = \frac{\sum_{\mathbf{x}\in\mathbf{D}} \mathbb{1}\{\mathbf{x} \models [i \land j]\}}{\sum_{\mathbf{x}\in\mathbf{D}} \mathbb{1}\{\mathbf{x} \models [i]\}}$$

samples activating node j

samples activating node i

$\Longrightarrow$ *global maximum with single pass over* $\mathbf{D}$

$\Longrightarrow$ *regularization, e.g. Laplace-smoothing, to avoid divide by zero*

$\Longrightarrow$ *when missing data, fallback to EM*

Kisa et al., "Probabilistic sentential decision diagrams", 2014
Peharz et al., "Learning Selective Sum-Product Networks", 2014
Liang et al., "Learning Logistic Circuits", 2019

Gradient descent

In alternative to EM, just descent the negative (log-)likelihood by (S)GD

$\Longrightarrow$ *circuits are differentiable!*

- ***backprop*** + your favorite gradient-based optimizer
- need to reparametrize sum node weights ... $\Longrightarrow$ *e.g. by (log-)softmax*
- ...or project them to their constraint set [Duchi2008]
- analogously for input distribution parameters

$\Longrightarrow$ *e.g.* $\sigma > 0$ *in Gaussians: use softplus or clipping*

Gradient descent

In alternative to EM, just descent the negative (log-)likelihood by (S)GD

$\Longrightarrow$ *circuits are differentiable!*

- ***backprop*** + your favorite gradient-based optimizer
- need to reparametrize sum node weights ... $\Longrightarrow$ *e.g. by (log-)softmax*
- ...or project them to their constraint set *[Duchi2008]*
- analogously for input distribution parameters

$\Longrightarrow$ *e.g.* $\sigma > 0$ *in Gaussians: use softplus or clipping*

pros:

- Easy to implement and combine
  with other cost functions

Gradient descent

In alternative to EM, just descent the negative (log-)likelihood by (S)GD

$\Longrightarrow$ *circuits are differentiable!*

- ***backprop*** + your favorite gradient-based optimizer
- need to reparametrize sum node weights ... $\Longrightarrow$ *e.g. by (log-)softmax*
- ...or project them to their constraint set *[Duchi2008]*
- analogously for input distribution parameters

$\Longrightarrow$ *e.g.* $\sigma > 0$ *in Gaussians: use softplus or clipping*

pros:

- Easy to implement and combine with other cost functions

cons:

- (S)GD converges slowly

Bayesian parameter learning

Formulate a prior $p(\mathbf{w}, \boldsymbol{\theta})$ over sum-weights and leaf-parameters and perform posterior inference:

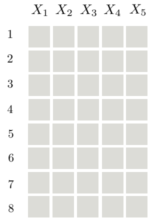
$$p(\mathbf{w}, \boldsymbol{\theta} | \mathcal{D}) \propto p(\mathbf{w}, \boldsymbol{\theta}) \, p(\mathcal{D} | \mathbf{w}, \boldsymbol{\theta})$$

- Moment matching (oBMM) *[Jaini et al. 2016; Rashwan et al. 2016]*
- Collapsed variational inference algorithm *[Zhao et al. 2016b]*
- Gibbs sampling *[Trapp et al. 2019; Vergari et al. 2019]*

Learning probabilistic circuits

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

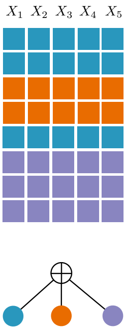
LearnSPN



Learning both structure and parameters of a circuit by starting from a data matrix

Gens et al., "Learning the Structure of Sum-Product Networks", 2013

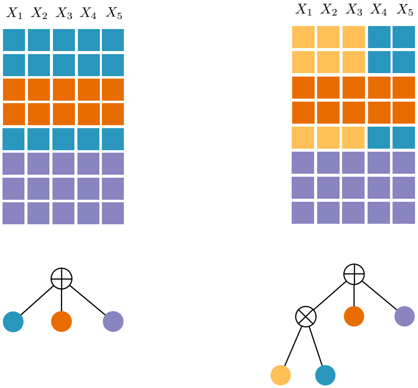
LearnSPN



Looking for sub-population in the data—***clustering***—to introduce sum nodes...

Gens et al., "Learning the Structure of Sum-Product Networks", 2013

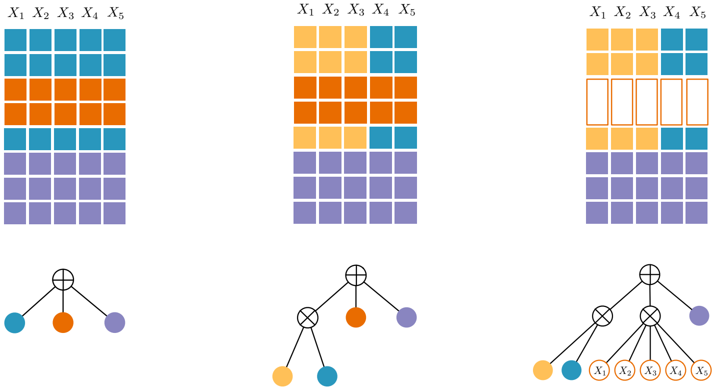
LearnSPN



...***seeking independencies among sets of RVs*** to factorize into product nodes

Gens et al., "Learning the Structure of Sum-Product Networks", 2013

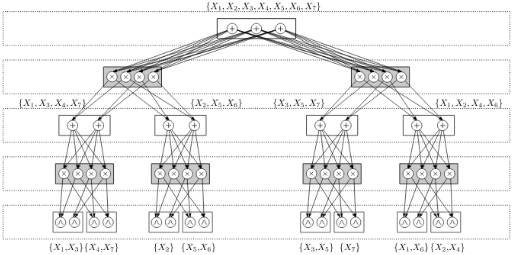
LearnSPN



...learning smaller estimators as a ***a recursive data crawler***

Gens et al., "Learning the Structure of Sum-Product Networks", 2013

Randomized structure learning



Randomly generate a region graph ⟹ *hierarchical partitioning of variables*

Then, populate each region with ***tensorized*** circuit nodes ⟹ *competitive with SOTA*

Peharz et al., "Random Sum-Product Networks: A Simple and Effective Approach to Probabilistic Deep Learning", 2019

Learning probabilistic circuits

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

Ensembles of probabilistic circuits

Single circuits might be not accurate enough or ***overfit*** training data...

Solution: *ensembles of circuits*!

$\Longrightarrow$ *non-deterministic mixture models: another sum node!*

$$p(\mathbf{X}) = \sum_{i=1}^{K} \lambda_i \mathcal{C}_i(\mathbf{X}), \quad \lambda_i \geq 0 \quad \sum_{i=1}^{K} \lambda_i = 1$$

Ensemble weights and components can be learned separately or jointly

- EM or structural EM *[Liang et al. 2017a]*
- bagging *[Vergari et al. 2015; Rahman et al. 2016; Di Mauro et al. 2017]*
- boosting *[Rahman et al. 2016]*

Learning probabilistic circuits

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

Applications

Stay tuned for...

Next:

1. what have been probabilistic circuits used for?

   ⇒ *computer vision, sop, speech, planning, ...*

2. what are the current trends in tractable learning?

   ⇒ *hybrid models, probabilistic programming, ...*

3. what are the current challenges?

   ⇒ *benchmarks, scaling, reasoning*

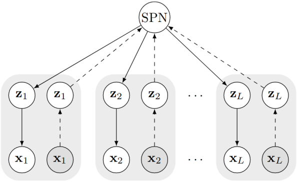
After: *Conclusions*

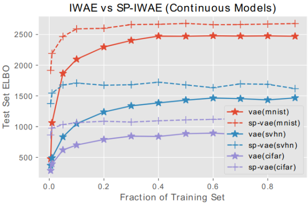
EVI inference : density estimation

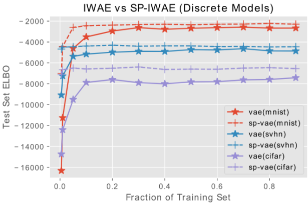
dataset	single models	ensembles	dataset	single models	ensembles
nltcs	-5.99 [ID-SPN]	-5.99 [LearnPSDDs]	dna	-79.88 [SPGM]	-80.07 [SPN-btb]
msnbc	-6.04 [Prometheus]	-6.04 [LearnPSDDs]	kosarek	-10.59 [Prometheus]	-10.52 [LearnPSDDs]
kdd	-2.12 [Prometheus]	-2.12 [LearnPSDDs]	msweb	-9.73 [ID-SPN]	-9.62 [XCNets]
plants	-12.54 [ID-SPN]	-11.84 [XCNets]	book	-34.14 [ID-SPN]	-33.82 [SPN-btb]
audio	-39.77 [BNP-SPN]	-39.39 [XCNets]	movie	-51.49 [Prometheus]	-50.34 [XCNets]
jester	-52.42 [BNP-SPN]	-51.29 [LearnPSDDs]	webkb	-151.84 [ID-SPN]	-149.20 [XCNets]
netflix	-56.36 [ID-SPN]	-55.71 [LearnPSDDs]	cr52	-83.35 [ID-SPN]	-81.87 [XCNets]
accidents	-26.89 [SPGM]	-29.10 [XCNets]	c20ng	-151.47 [ID-SPN]	-151.02 [XCNets]
retail	-10.85 [ID-SPN]	-10.72 [LearnPSDDs]	bbc	-248.5 [Prometheus]	-229.21 [XCNets]
pumbs*	-22.15 [SPGM]	-22.67 [SPN-btb]	ad	-15.40 [CNetXD]	-14.00 [XCNets]

Hybrid intractable* + *tractable EVI

VAEs as intractable input distributions, orchestrated by a circuit on top





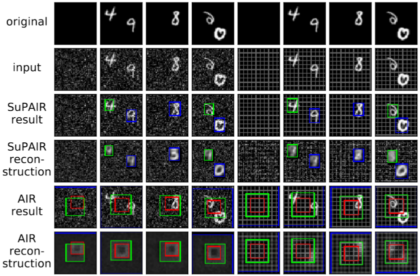


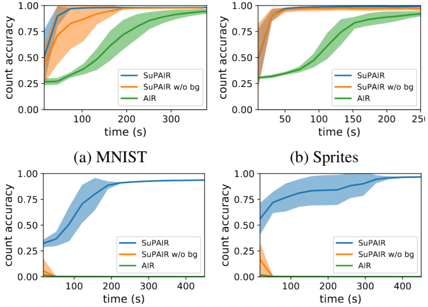
$\Rightarrow$ *decomposing a joint ELBO: better lower-bounds than a single VAE*

$\Rightarrow$ *more expressive efficient and less data hungry*

Tan et al., "Hierarchical Decompositional Mixtures of Variational Autoencoders", 2019

Tractable MAR : scene understanding





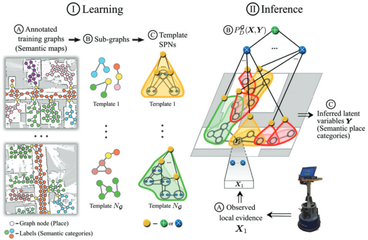
(a) MNIST (b) Sprites

(c) Noisy MNIST (d) Grid MNIST

⇒ *making the AIR model faster and more accurate by using a PC*

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

Tractable MAR : Robotics



Hierarchical planning robot executions

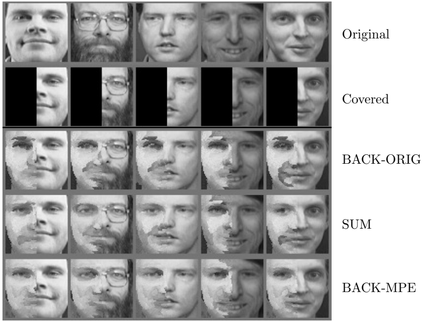
Scenes and maps decompose along circuit structures

Pronobis et al., "Learning Deep Generative Spatial Models for Mobile Robots", 2016

Pronobis et al., "Deep spatial affordance hierarchy: Spatial knowledge representation for planning in large-scale environments", 2017

Zheng et al., "Learning graph-structured sum-product networks for probabilistic semantic maps", 2018

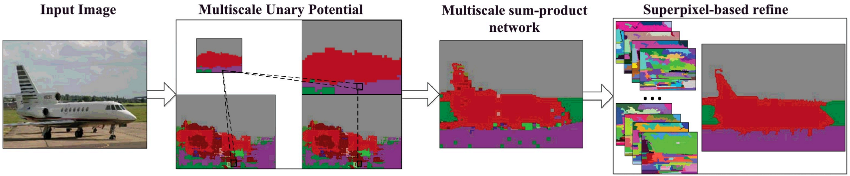
MAP inference : **image inpainting**



Predicting ***arbitrary patches***
given a ***single*** circuit
First SPN paper in 2011...

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

MAP inference : image segmentation



Semantic segmentation is MAP over joint pixel and label space

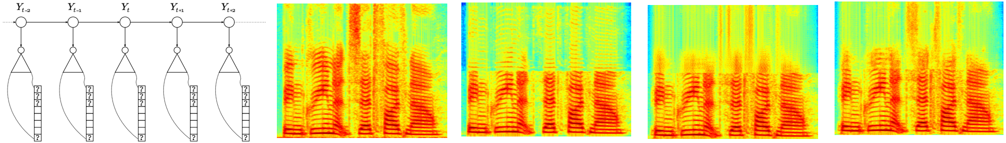
Even approximate MAP for non-deterministic circuits (SPNs) delivers good performances.

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

MAP inference : Speech reconstruction

Probabilistic circuits to model the joint pdf of ***observables in HMMs*** (HMM-SPNs),

again leveraging tractable inference: MAR and MAP



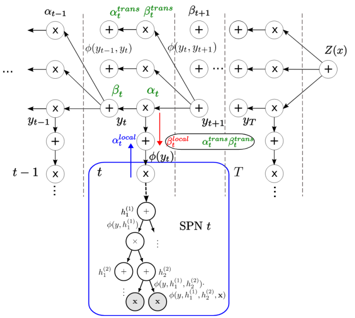
(a) Original full bandwidth (b) Reconstruction HMM-LP (c) Reconstruction HMM-GMM (d) Reconstruction HMM-SPN

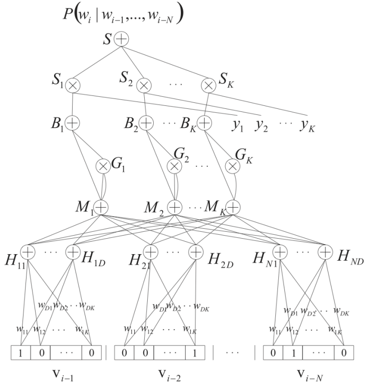
State-of-the-art high frequency reconstruction (MAP inference)

Peharz et al., "Modeling speech with sum-product networks: Application to bandwidth extension", 2014

Zohrer et al., "Representation learning for single-channel source separation and bandwidth extension", 2015

MAP inference : Sequence labeling

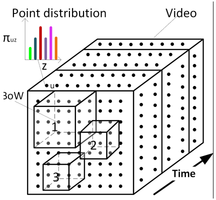


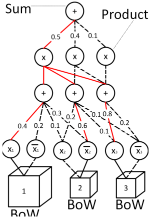


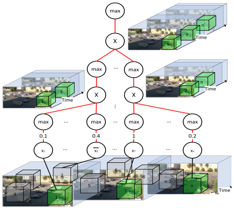
Ratajczak et al., “Sum-Product Networks for Structured Prediction: Context-Specific Deep Conditional Random Fields”, 2014
Ratajczak et al., “Sum-Product Networks for Sequence Labeling”, 2018
Cheng et al., “Language modeling with Sum-Product Networks”, 2014

MAP and MMAP : activity recognition

Exploiting part-based decomposability along pixels *and time* (frames).

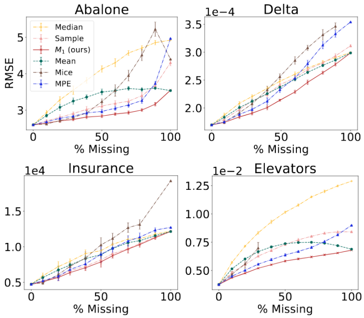






Amer et al., "Sum Product Networks for Activity Recognition", 2015
Wang et al., "Hierarchical spatial sum–product networks for action recognition in still images", 2016
Chiradeep Roy et al., "Explainable Activity Recognition in Videos using Dynamic Cutset Networks", 2019

ADV inference : expected predictions



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

$\Longrightarrow$ *missing values at test time*

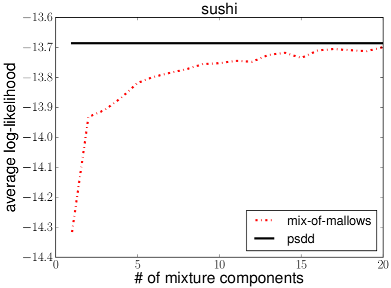
$\Longrightarrow$ *exploratory classifier analysis*

$$\mathbb{E}_{\mathbf{x}^m \sim p_\theta(\mathbf{x}^m|\mathbf{x}^o)} \left[f_\phi^k(\mathbf{x}^m, \mathbf{x}^o)\right]$$

Closed form moments for $\boldsymbol{f}$ and $\boldsymbol{p}$ as structured decomposable circuits with same v-tree

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

ADV inference : preference learning



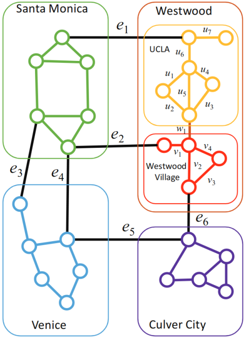
Preferences and rankings as logical constraints

Structured decomposable circuits for inference over structured spaces

SOTA on modeling densities over rankings

Choi et al., "Tractable learning for structured probability spaces: A case study in learning preference distributions", 2015
Shen et al., "A Tractable Probabilistic Model for Subset Selection.", 2017

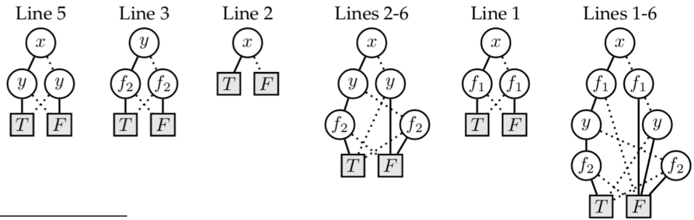
ADV inference : routing



Decomposing complex (conditional) probability spaces

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

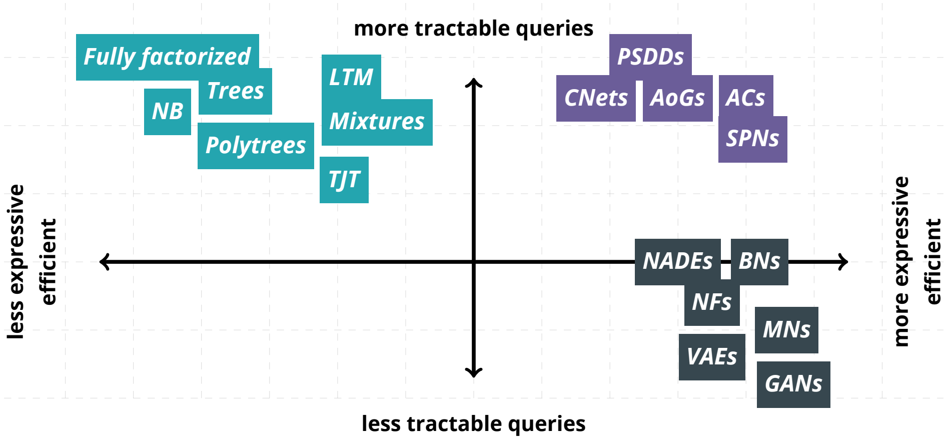
De Raedt et al.; Riguzzi; Fierens et al.; Vlasselaer et al., "ProbLog: A Probabilistic Prolog and Its Application in Link Discovery."; "A top down interpreter for LPAD and CP-logic"; "Inference and Learning in Probabilistic Logic Programs using Weighted Boolean Formulas"; "Anytime Inference in Probabilistic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

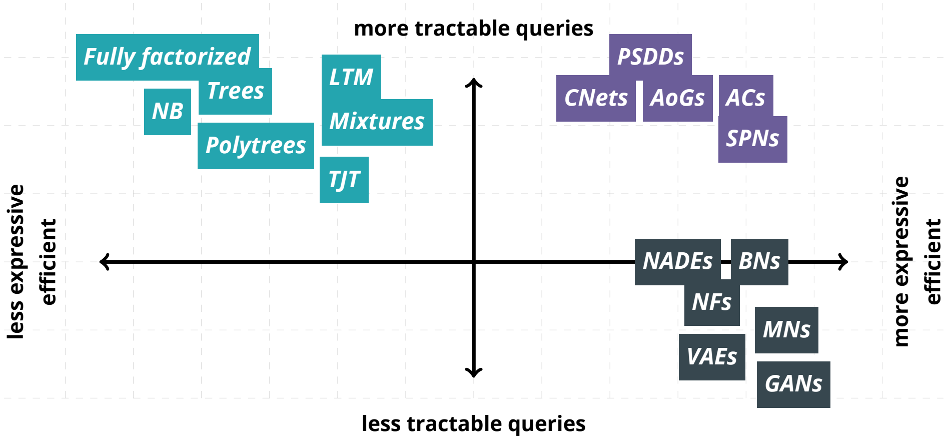
Vlasselaer et al., "Exploiting Local and Repeated Structure in Dynamic Bayesian Networks", 2016

and more...

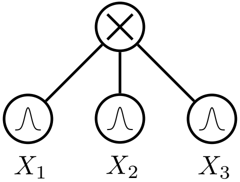
fault prediction *[Nath et al. 2016]*
computational psychology *[Joshi et al. 2018]*
biology *[Butz et al. 2018]*
low-energy prediction *[Galindez Olascoaga et al. 2019; Shah et al. 2019]*
calibration of analog/RF circuits *[Andraud et al. 2018]*
stochastic constraint optimization *[Latour et al. 2017]*
neuro-symbolic learning *[Xu et al. 2018]*
probabilistic and symbolic reasoning integration *[Li 2015]*
relational learning *[Broeck et al. 2011; Domingos et al. 2012; Broeck 2013; Nath et al. 2014, 2015; Niepert et al. 2015; Van Haaren et al. 2015]*

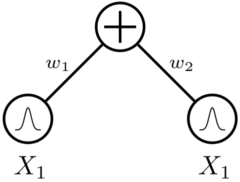


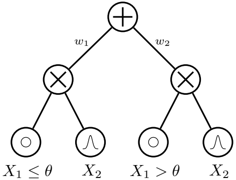
takeaway #1 tractability is a spectrum



takeaway #2: you can be both tractable and expressive







takeaway #3: probabilistic circuits are a foundation for tractable inference and learning

Challenge #1

hybridizing tractable and intractable models

Hybridize probabilistic inference*:*

tractable models inside intractable loops

and intractable small boxes glued by tractable inference!

Challenge #2

scaling tractable learning

Learn tractable models
on **millions of datapoints**
and **thousands of features**
in tractable time!

Challenge #3

advanced and automated reasoning

Move beyond single probabilistic queries
towards **fully automated reasoning**!

more links

`github.com/arranger1044/awesome-spn`

Libraries

Juice.jl a library for advanced logical and probabilistic inference with circuits in Julia ***SOON!***

SPFlow easy and extensible python library for SPNs `github.com/SPFlow/SPFlow`

Libra structure learning algorithms in OCaml `libra.cs.uoregon.edu`

Can your VAE inpaint any pixel patch?

Can your Flow flawles deal with missing values?

Can you obtain calibrated uncertainties from your GAN?

Join the discussion on the ***current state*** *of* ***probabilistic inference and learning*** *at the first*

11 Dec. 2019 from ***7***pm
Room 223-224
NeurIPS 2019*, Vancouver*

t'! tractable probabilistic inference meeting!

sites.google.com/view/tprime2019

relationalAI Uber

References I

- 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.
- Meilă, Marina and Michael I. Jordan (2000). "Learning with mixtures of trees". In: *Journal of Machine Learning Research* 1, pp. 1–48.
- 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.
- Chickering, Max (2002). "The WinMine Toolkit". In: *Microsoft, Redmond*.
- Darwiche, Adnan and Pierre Marquis (2002). "A knowledge compilation map". In: *Journal of Artificial Intelligence Research* 17, pp. 229–264.

References II

- 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*.
- Sang, Tian, Paul Beame, and Henry A Kautz (2005). "Performing Bayesian inference by weighted model counting". In: *AAAI*. Vol. 5, pp. 475–481.
- 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.
- 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.
- Kulesza, A. and F. Pereira (2007). "Structured Learning with Approximate Inference". In: *Advances in Neural Information Processing Systems 20*. MIT Press, pp. 785–792.
- 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.
- Lowd, Daniel and Pedro Domingos (2008). "Learning Arithmetic Circuits". In: *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*. UAI'08. Helsinki, Finland: AUAI Press, pp. 383–392. ISBN: 0-9749039-4-9. URL: `http://dl.acm.org/citation.cfm?id=3023476.3023522`.

References III

- 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.
- 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.
- Lowd, Daniel and Pedro Domingos (2010). “Approximate inference by compilation to arithmetic circuits”. In: *Advances in Neural Information Processing Systems*, pp. 1477–1485.
- Broeck, Guy Van den et al. (2011). “Lifted probabilistic inference by first-order knowledge compilation”. In: *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*. AAAI Press/International Joint Conferences on Artificial Intelligence; Menlo ..., pp. 2178–2185.
- Campos, Cassio Polpo de (2011). “New complexity results for MAP in Bayesian networks”. In: *IJCAI*. Vol. 11, pp. 2100–2106.
- Larochelle, Hugo and Iain Murray (2011). “The Neural Autoregressive Distribution Estimator”. In: *International Conference on Artificial Intelligence and Statistics*, pp. 29–37.
- 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.
- Domingos, Pedro and William Austin Webb (2012). “A tractable first-order probabilistic logic”. In: *Twenty-Sixth AAAI Conference on Artificial Intelligence*.
- Gens, Robert and Pedro Domingos (2012). “Discriminative Learning of Sum-Product Networks”. In: *Advances in Neural Information Processing Systems 25*, pp. 3239–3247.

References IV

- Broeck, Guy Van den (2013). “Lifted inference and learning in statistical relational models”. PhD thesis. Ph. D. Dissertation, KU Leuven.
- 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.
- Peharz, Robert, Bernhard Geiger, and Franz Pernkopf (2013). “Greedy Part-Wise Learning of Sum-Product Networks”. In: *ECML-PKDD 2013*.
- Cheng, Wei-Chen et al. (2014). “Language modeling with Sum-Product Networks”. In: *INTERSPEECH 2014*, pp. 2098–2102.
- Goodfellow, Ian et al. (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 et al. (July 2014a). “Probabilistic sentential decision diagrams”. In: *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*. Vienna, Austria.
- — (July 2014b). “Probabilistic sentential decision diagrams”. In: *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*. Vienna, Austria. URL: `http://starai.cs.ucla.edu/papers/KisaKR14.pdf`.
- Martens, James and Venkatesh Medabalimi (2014). “On the Expressive Efficiency of Sum Product Networks”. In: *CoRR* abs/1411.7717.
- Nath, Aniruddh and Pedro Domingos (2014). “Learning Tractable Statistical Relational Models”. In: *Workshop on Learning Tractable Probabilistic Models, ICML 2014*.

References V

- Peharz, Robert, Robert Gens, and Pedro Domingos (2014a). “Learning Selective Sum-Product Networks”. In: *Workshop on Learning Tractable Probabilistic Models*. LTPM.
- Peharz, Robert et al. (2014b). “Modeling speech with sum-product networks: Application to bandwidth extension”. In: *ICASSP2014*.
- 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.
- Ratajczak, Martin, S Tschiatschek, and F Pernkopf (2014). “Sum-Product Networks for Structured Prediction: Context-Specific Deep Conditional Random Fields”. In: *Proc Workshop on Learning Tractable Probabilistic Models* 1, pp. 1–10.
- Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). “Stochastic backprop. and approximate inference in deep generative models”. In: *arXiv preprint arXiv:1401.4082*.
- Rooshenas, Amirmohammad and Daniel Lowd (2014). “Learning Sum-Product Networks with Direct and Indirect Variable Interactions”. In: *Proceedings of ICML 2014*.
- Amer, Mohamed and Sinisa Todorovic (2015). “Sum Product Networks for Activity Recognition”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on*.
- Bekker, Jessa et al. (2015). “Tractable Learning for Complex Probability Queries”. In: *Advances in Neural Information Processing Systems 28 (NIPS)*.
- Burda, Yuri, Roger Grosse, and Ruslan Salakhutdinov (2015). “Importance weighted autoencoders”. In: *arXiv preprint arXiv:1509.00519*.
- Choi, Arthur, Guy Van den Broeck, and Adnan Darwiche (2015). “Tractable learning for structured probability spaces: A case study in learning preference distributions”. In: *Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*.

References VI

- 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.
- Fierens, Daan et al. (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 et al. (2015). “MADE: Masked Autoencoder for Distribution Estimation”. In: *CoRR* abs/1502.03509.
- Li, Weizhuo (2015). “Combining sum-product network and noisy-or model for ontology matching.”. In: *OM*, pp. 35–39.
- Nath, Aniruddh and Pedro Domingos (2015). “Learning Relational Sum-Product Networks”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*.
- Niepert, Mathias and Pedro Domingos (2015). “Learning and inference in tractable probabilistic knowledge bases”. In: *AUAI Press*.
- Peharz, Robert (2015). “Foundations of Sum-Product Networks for Probabilistic Modeling”. PhD thesis. Graz University of Technology, SPSC.
- Peharz, Robert et al. (2015). “On Theoretical Properties of Sum-Product Networks”. In: *The Journal of Machine Learning Research*.
- Van Haaren, Jan et al. (2015). “Lifted Generative Learning of Markov Logic Networks”. In: *Machine Learning* 103.1, pp. 27–55. DOI: 10.1007/s10994-015-5532-x.
- Vergari, Antonio, Nicola Di Mauro, and Floriana Esposito (2015). “Simplifying, Regularizing and Strengthening Sum-Product Network Structure Learning”. In: *ECML-PKDD 2015*.
- Vlasselaer, Jonas et al. (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.

References VII

- Zhao, Han, Mazen Melibari, and Pascal Poupart (2015). “On the Relationship between Sum-Product Networks and Bayesian Networks”. In: *ICML*.
- Zohrer, Matthias, Robert Peharz, and Franz Pernkopf (2015). “Representation learning for single-channel source separation and bandwidth extension”. In: *Audio, Speech, and Language Processing, IEEE/ACM Transactions on* 23.12, pp. 2398–2409.
- 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.
- Friesen, Abram L and Pedro Domingos (2016). “Submodular Sum-product Networks for Scene Understanding”. In:
- Jaini, Priyank et al. (2016). “Online Algorithms for Sum-Product Networks with Continuous Variables”. In: *Probabilistic Graphical Models - Eighth International Conference, PGM 2016, Lugano, Switzerland, September 6-9, 2016. Proceedings*, pp. 228–239. URL: `http://jmlr.org/proceedings/papers/v52/jaini16.html`.
- Nath, Aniruddh and Pedro M. Domingos (2016). “Learning Tractable Probabilistic Models for Fault Localization”. In: *CoRR* abs/1507.01698. URL: `http://arxiv.org/abs/1507.01698`.
- 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 et al. (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`.
- Pronobis, A. and R. P. N. Rao (2016). “Learning Deep Generative Spatial Models for Mobile Robots”. In: *ArXiv e-prints*. arXiv: `1610.02627 [cs.RO]`.

References VIII

- Rahman, Tahrima and Vibhav Gogate (2016). “Learning Ensembles of Cutset Networks”. In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI’16. Phoenix, Arizona: AAAI Press, pp. 3301–3307. URL: `http://dl.acm.org/citation.cfm?id=3016100.3016365`.
- Rashwan, Abdullah, Han Zhao, and Pascal Poupart (2016). “Online and Distributed Bayesian Moment Matching for Parameter Learning in Sum-Product Networks”. In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pp. 1469–1477.
- Rooshenas, Amirmohammad and Daniel Lowd (2016). “Discriminative Structure Learning of Arithmetic Circuits”. In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pp. 1506–1514.
- 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.
- Sharir, Or et al. (2016). “Tractable generative convolutional arithmetic circuits”. In: *arXiv preprint arXiv:1610.04167*.
- 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 et al. (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`.
- Wang, Jinghua and Gang Wang (2016). “Hierarchical spatial sum–product networks for action recognition in still images”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 28.1, pp. 90–100.
- Yuan, Zehuan et al. (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

- Zhao, Han, Pascal Poupart, and Geoffrey J Gordon (2016a). "A Unified Approach for Learning the Parameters of Sum-Product Networks". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., pp. 433–441.
- Zhao, Han et al. (2016b). "Collapsed Variational Inference for Sum-Product Networks". In: *In Proceedings of the 33rd International Conference on Machine Learning*. Vol. 48.
- Alemi, Alexander A et al. (2017). "Fixing a broken ELBO". In: *arXiv preprint arXiv:1711.00464*.
- 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.
- Di Mauro, Nicola et al. (2017). "Fast and Accurate Density Estimation with Extremely Randomized Cutset Networks". In: *ECML-PKDD 2017*.
- Latour, Anna et al. (Aug. 2017). "Combining Stochastic Constraint Optimization and Probabilistic Programming: From Knowledge Compilation to Constraint Solving". In: *Proceedings of the 23rd International Conference on Principles and Practice of Constraint Programming (CP)*. DOI: `10.1007/978-3-319-66158-2_32`.
- Liang, Yitao, Jessa Bekker, and Guy Van den Broeck (2017a). "Learning the structure of probabilistic sentential decision diagrams". In: *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*.
- Liang, Yitao and Guy Van den Broeck (Aug. 2017b). "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`.

References X

- Pronobis, Andrzej, Francesco Riccio, and Rajesh PN Rao (2017). “Deep spatial affordance hierarchy: Spatial knowledge representation for planning in large-scale environments”. In: *ICAPS 2017 Workshop on Planning and Robotics, Pittsburgh, PA, USA*.
- 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.
- Salimans, Tim et al. (2017). “PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications”. In: *arXiv preprint arXiv:1701.05517*.
- Shen, Yujia, Arthur Choi, and Adnan Darwiche (2017). “A Tractable Probabilistic Model for Subset Selection.”. In: *UAI*.
- 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.
- Andraud, Martin et al. (2018). “On the use of Bayesian Networks for Resource-Efficient Self-Calibration of Analog/RF ICs”. In: *2018 IEEE International Test Conference (ITC)*. IEEE, pp. 1–10.
- Butz, Cory J et al. (2018). “Efficient Examination of Soil Bacteria Using Probabilistic Graphical Models”. In: *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, pp. 315–326.
- Choi, YooJung and Guy Van den Broeck (2018). “On robust trimming of Bayesian network classifiers”. In: *arXiv preprint arXiv:1805.11243*.
- Friedman, Tal and Guy Van den Broeck (Dec. 2018). “Approximate Knowledge Compilation by Online Collapsed Importance Sampling”. In: *Advances in Neural Information Processing Systems 31 (NeurIPS)*. URL: http://starai.cs.ucla.edu/papers/FriedmanNeurIPS18.pdf.

References XI

- Joshi, Himanshu, Paul S Rosenbloom, and Volkan Ustun (2018). “Exact, tractable inference in the Sigma cognitive architecture via sum-product networks”. In: *Advances in Cognitive Systems*.
- Rashwan, Abdullah, Pascal Poupart, and Chen Zhitang (2018). “Discriminative Training of Sum-Product Networks by Extended Baum-Welch”. In: *International Conference on Probabilistic Graphical Models*, pp. 356–367.
- Ratajczak, Martin, Sebastian Tschiatschek, and Franz Pernkopf (2018). “Sum-Product Networks for Sequence Labeling”. In: *arXiv preprint arXiv:1807.02324*.
- Shen, Yujia, Arthur Choi, and Adnan Darwiche (2018). “Conditional PSDDs: Modeling and learning with modular knowledge”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*.
- Xu, Jingyi et al. (July 2018). “A Semantic Loss Function for Deep Learning with Symbolic Knowledge”. In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*.
- Zheng, Kaiyu, Andrzej Pronobis, and Rajesh PN Rao (2018). “Learning graph-structured sum-product networks for probabilistic semantic maps”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*.
- Chiradeep Roy, Tahrima Rahman and Vibhav Gogate (2019). “Explainable Activity Recognition in Videos using Dynamic Cutset Networks”. In: *TPM2019*.
- Dai, Bin and David Wipf (2019). “Diagnosing and enhancing vae models”. In: *arXiv preprint arXiv:1903.05789*.
- Galindez Olascoaga, Laura Isabel et al. (2019). “Towards Hardware-Aware Tractable Learning of Probabilistic Models”. In: *Proceedings of the ICML Workshop on Tractable Probabilistic Modeling (TPM)*. URL: `http://starai.cs.ucla.edu/papers/GalindezTPM19.pdf`.
- Ghosh, Partha et al. (2019). “From variational to deterministic autoencoders”. In: *arXiv preprint arXiv:1903.12436*.

References XII

- 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 et al. (2019a). "On Tractable Computation of Expected Predictions". In: *Advances in Neural Information Processing Systems*, pp. 11167–11178.
- Khosravi, Pasha et al. (2019b). "What to Expect of Classifiers? Reasoning about Logistic Regression with Missing Features". In: *arXiv preprint arXiv:1903.01620*.
- Khosravi, Pasha et al. (2019c). "What to Expect of Classifiers? Reasoning about Logistic Regression with Missing Features". In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*.
- Kossen, Jannik et al. (2019). "Structured Object-Aware Physics Prediction for Video Modeling and Planning". In: *arXiv preprint arXiv:1910.02425*.
- Liang, Yitao and Guy Van den Broeck (2019). "Learning Logistic Circuits". In: *Proceedings of the 33rd Conference on Artificial Intelligence (AAAI)*.
- Peharz, Robert et al. (2019). "Random Sum-Product Networks: A Simple and Effective Approach to Probabilistic Deep Learning". In: *Uncertainty in Artificial Intelligence*.
- Shah, Nimish et al. (2019). "ProbLP: A framework for low-precision probabilistic inference". In: *Proceedings of the 56th Annual Design Automation Conference 2019*. ACM, p. 190.
- Shao, Xiaoting et al. (2019). "Conditional Sum-Product Networks: Imposing Structure on Deep Probabilistic Architectures". In: *arXiv preprint arXiv:1905.08550*.
- Shen, Yujia et al. (2019). "Structured Bayesian Networks: From Inference to Learning with Routes". In: *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*.
- Shih, Andy et al. (2019). "Smoothing Structured Decomposable Circuits". In: *arXiv preprint arXiv:1906.00311*.

References XIII

- 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.
- Tan, Ping Liang and Robert Peharz (2019). “Hierarchical Decompositional Mixtures of Variational Autoencoders”. 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. 6115–6124. URL: http://proceedings.mlr.press/v97/tan19b.html.
- Trapp, Martin et al. (2019). “Bayesian Learning of Sum-Product Networks”. In: *Advances in neural information processing systems (NeurIPS)*.
- Vergari, Antonio et al. (2019). “Automatic Bayesian density analysis”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33, pp. 5207–5215.