Tractable Probabilistic Models

Representations
Inference
Learning
Applications

Guy Van den Broeck
University of California, Los Angeles

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

Antonio Vergari
University of California, Los Angeles

YooJung Choi
University of California, Los Angeles

Robert Peharz
TU Eindhoven

Nicola Di Mauro
University of Bari

*November 11, 2019 - **International Spring School on "Uncertainty in AI and data management"*** **- Santiago, Chile**

AoGs | PDGs | NBs | Fully factorized | sd-DNNF | PSDDs

Trees | LTMs | DNNFs | OBDDs | CNets | SPNs | NADEs

Thin Junction Trees | NNF | FBDDs | BDDs | ACs | VAEs

Polytrees | d-NNFs | ADDs | SDDs | TACs | GANs | NFs

Mixtures | XADDs | XSDDs | MNs | BNs | FGs

The Alphabet Soup of models in AI

AoGs PDGs NBs Fully factorized sd-DNNF PSDDs

Trees LTMs DNNFs OBDDs CNets SPNs NADEs

Thin Junction Trees NNF FBDDs BDDs ACs VAEs

Polytrees d-NNFs ADDs SDDs TACs GANs NFs

Mixtures XADDs XSDDs MNs BNs FGs

Logical and *Probabilistic* models

AoGs *PDGs* *NBs* *Fully factorized* *sd-DNNF* *PSDDs*

Trees *LTMs* *DNNFs* *OBDDs* *CNets* *SPNs* *NADEs*

Thin Junction Trees *NNF* *FBDDs* *BDDs* *ACs* *VAEs*

Polytrees *d-NNFs* *ADDs* *SDDs* *TACs* *GANs* *NFs*

Mixtures *XADDs* *XSDDs* *MNs* *BNs* *FGs*

Tractable and *Intractable* probabilistic models

AoGs PDGs NBs Fully factorized sd-DNNF PSDDs

Trees LTMs DNNFs OBDDs CNets SPNs NADEs

Thin Junction Trees NNF FBDDs BDDs ACs VAEs

Polytrees d-NNFs ADDs SDDs TACs GANs NFs

Mixtures XADDs XSDDs MNs BNs FGs

Expressive **models without** *compromises*

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

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 Herzl Str.?*

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

pinterest.com/pin/190417890473268205/

Why probabilistic inference?

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

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

$\Longrightarrow$ fitting a predictive model!

pinterest.com/pin/190417890473268205/

Why probabilistic inference?

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

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

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

$\Longrightarrow$ answering probabilistic ***queries*** on a probabilistic model of the world $\mathbf{m}$

$$\mathbf{q}_1(\mathbf{m}) = \textbf{?} \qquad \mathbf{q}_2(\mathbf{m}) = \textbf{?}$$

pinterest.com/pin/190417890473268205/

Why probabilistic inference?

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

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

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

pinterest.com/pin/190417890473268205/

Why probabilistic inference?

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

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

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

$\Longrightarrow$ ***marginals***

pinterest.com/pin/190417890473268205/

Why probabilistic inference?

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

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

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

pinterest.com/pin/190417890473268205/

Why probabilistic inference?

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

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

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

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

pinterest.com/pin/190417890473268205/

Tractable Probabilistic Inference

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

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

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

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

$\implies$ *think of* $|\mathbf{m}|$ *as the number of streets on my route to work*

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

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

$\Longrightarrow$ *think of* $|\mathbf{m}|$ *as the number of streets on my route to work*

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

What about approximate inference?

- Why approximate when we can do exact?

  $\Longrightarrow$ *and do we lose some expressiveness?*

- Approximations can be intractable as well *[Dagum et al. 1993; Roth 1996]*

  $\Longrightarrow$ *But sometimes approximate inference comes with guarantees*

- Approximate inference by exact inference in approximate model

  [Dechter et al. 2002; Choi et al. 2010; Lowd et al. 2010; Sontag et al. 2011; Friedman et al. 2018]

- Approximate inference (even with guarantees) can mislead learners

  [Kulesza et al. 2007] $\Longrightarrow$ *Chaining approximations is flying with a blindfold on*

What about approximate inference?

- Why approximate when we can do exact?

  ⟹ *and do we lose some expressiveness?*

- Approximations can be intractable as well *[Dagum et al. 1993; Roth 1996]*

  ⟹ *But sometimes approximate inference comes with guarantees*

- Approximate inference by exact inference in approximate model

  [Dechter et al. 2002; Choi et al. 2010; Lowd et al. 2010; Sontag et al. 2011; Friedman et al. 2018]

- Approximate inference (even with guarantees) can mislead learners

  [Kulesza et al. 2007] ⟹ *Chaining approximations is flying with a blindfold on*

What about approximate inference?

- Why approximate when we can do exact?

  $\Longrightarrow$ *and do we lose some expressiveness?*

- Approximations can be intractable as well *[Dagum et al. 1993; Roth 1996]*

  $\Longrightarrow$ *But sometimes approximate inference comes with guarantees*

- Approximate inference by exact inference in approximate model

  [Dechter et al. 2002; Choi et al. 2010; Lowd et al. 2010; Sontag et al. 2011; Friedman et al. 2018]

- Approximate inference (even with guarantees) can mislead learners

  [Kulesza et al. 2007] $\Longrightarrow$ *Chaining approximations is flying with a blindfold on*

What about approximate inference?

- Why approximate when we can do exact?

  $\Longrightarrow$ *and do we lose some expressiveness?*

- Approximations can be intractable as well *[Dagum et al. 1993; Roth 1996]*

  $\Longrightarrow$ *But sometimes approximate inference comes with guarantees*

- Approximate inference by exact inference in approximate model

  [Dechter et al. 2002; Choi et al. 2010; Lowd et al. 2010; Sontag et al. 2011; Friedman et al. 2018]

- Approximate inference (even with guarantees) can mislead learners

  [Kulesza et al. 2007] $\Longrightarrow$ *Chaining approximations is flying with a blindfold on*

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 queries (EVI)

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

pinterest.com/pin/190417890473268205/

Complete evidence queries (EVI)

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

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

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

pinterest.com/pin/190417890473268205/

Complete evidence queries (EVI)

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

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

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

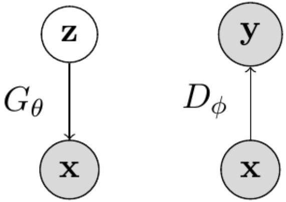
...fundamental in ***maximum likelihood learning***

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

pinterest.com/pin/190417890473268205/

Generative Adversarial Networks

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

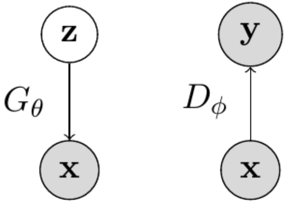


Goodfellow et al., "Generative adversarial nets", 2014

~~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*

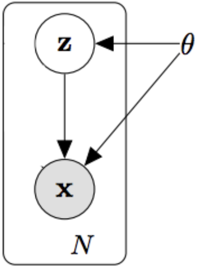


Goodfellow et al., "Generative adversarial nets", 2014

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~~

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

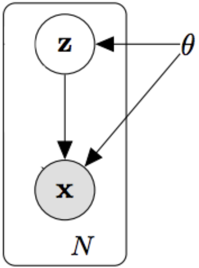
- an explicit likelihood model!
- ... but computing $\log p_\theta(\mathbf{x})$ is intractable

  $\Longrightarrow$ *an infinite and uncountable mixture*

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

  $\Longrightarrow$ *which is "broken"*

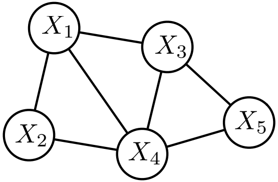
  [Alemi et al. 2017; Dai et al. 2019]



Probabilistic Graphical Models (PGMs)

Declarative semantics: a clean separation of modeling assumptions from inference

Nodes: random variables
Edges: dependencies



+

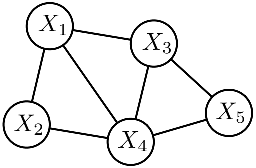
Inference:

- 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]*

PGMs: MNs and BNs

Markov Networks (MNs)

$p(\mathbf{X}) = \frac{1}{Z} \prod_c \phi_c(\mathbf{X}_c)$



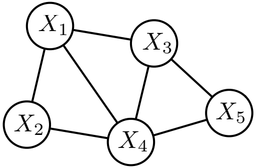
PGMs: MNs and BNs

Markov Networks (MNs)

$$p(\mathbf{X}) = \frac{1}{Z} \prod_c \phi_c(\mathbf{X}_c)$$

$$Z = \int \prod_c \phi_c(\mathbf{X}_c) d\mathbf{X}$$

$\Longrightarrow$ EVI *queries are intractable!*



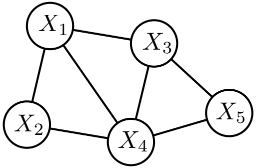
PGMs: MNs and BNs

Markov Networks (MNs)

$$p(\mathbf{X}) = \frac{1}{Z} \prod_c \phi_c(\mathbf{X}_c)$$

$$Z = \int \prod_c \phi_c(\mathbf{X}_c) d\mathbf{X}$$

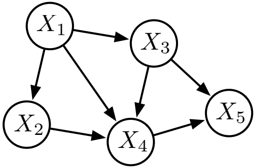
$\Longrightarrow$ EVI *queries are intractable!*



Bayesian Networks (BNs)

$$p(\mathbf{X}) = \prod_i p(X_i \mid \mathsf{pa}(X_i))$$

$\Longrightarrow$ EVI *queries are tractable!*



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 Herzl Str.?*

pinterest.com/pin/190417890473268205/

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 Herzl Str.?*

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

`pinterest.com/pin/190417890473268205/`

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 Herzl Str.?*

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

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

where $\mathbf{E} \subset \mathbf{X}$

$$\mathbf{H} = \mathbf{X} \setminus \mathbf{E}$$

pinterest.com/pin/190417890473268205/

Conditional queries (CON)

q_4: *What is the probability that there is a traffic jam on Herzl Str.* ***given that*** *today is a Monday?*

pinterest.com/pin/190417890473268205/

Conditional queries (CON)

$\mathbf{q}_4$: *What is the probability that there is a traffic jam on Herzl Str.* ***given that*** *today is a Monday?*

$$\mathbf{q}_4(\mathbf{m}) = p_{\mathbf{m}}(\mathsf{Jam}_{\mathsf{Herzl}} = 1 \mid \mathsf{Day} = \mathsf{Mon})$$

pinterest.com/pin/190417890473268205/

Conditional queries (CON)

$\mathbf{q}_4$: *What is the probability that there is a traffic jam on Herzl Str.* ***given that*** *today is a Monday?*

$$\mathbf{q}_4(\mathbf{m}) = p_{\mathbf{m}}(\mathsf{Jam}_{\mathsf{Herzl}} = 1 \mid \mathsf{Day} = \mathsf{Mon})$$

If you can answer MAR queries,
then you can also do ***conditional queries*** (CON):

$$p_{\mathbf{m}}(\mathbf{Q} \mid \mathbf{E}) = \frac{p_{\mathbf{m}}(\mathbf{Q}, \mathbf{E})}{p_{\mathbf{m}}(\mathbf{E})}$$

pinterest.com/pin/190417890473268205/

Complexity of MAR on PGMs

Exact complexity: Computing MAR and COND is *#P-complete* [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* [Dagum et al. 1993; Roth 1996].

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?*

Complexity of MAR on PGMs

Exact complexity: Computing MAR and COND is *#P-complete* *[Cooper 1990; Roth 1996].*

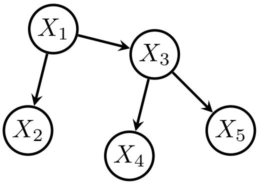
Approximation complexity: Computing MAR and COND approximately within a relative error of $2^{n^{1-\epsilon}}$ for any fixed ϵ is *NP-hard* *[Dagum et al. 1993; Roth 1996].*

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].*

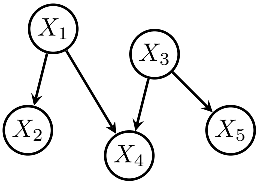
$\Longrightarrow$ *what about bounding the treewidth by design?*

Low-treewidth PGMs



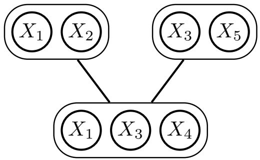
Trees

[Meilă et al. 2000]



Polytrees

[Dasgupta 1999]



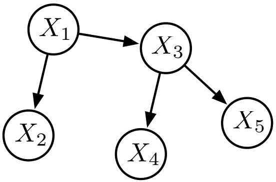
Thin Junction trees

[Bach et al. 2001]

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

Low-treewidth PGMs: trees

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



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

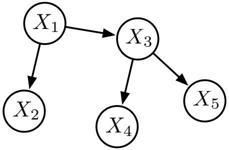
Exact querying: EVI, MAR, CON tasks *linear* for trees: $O(|\mathbf{X}|)$

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

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

What do we lose?

Expressiveness: Ability to compactly represent rich and complex classes of distributions

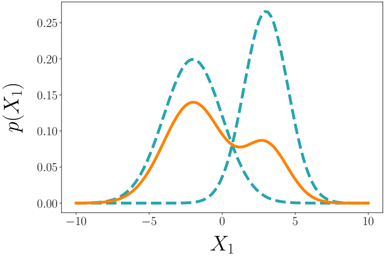


Bounded-treewidth PGMs lose the ability to represent *all possible distributions* ...

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

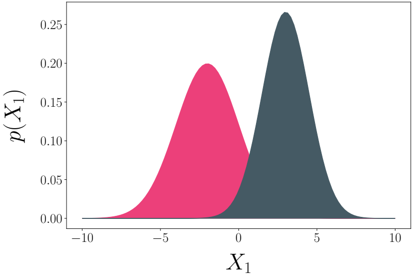


$$p(X) = w_1 \cdot p_1(X) + w_2 \cdot p_2(X)$$

EVI, MAR, CON queries scale linearly in k

Mixtures

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



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

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

$\Rightarrow$ *increased expressiveness*

Expressiveness and efficiency

Expressiveness: Ability to compactly represent rich and effective classes of functions

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

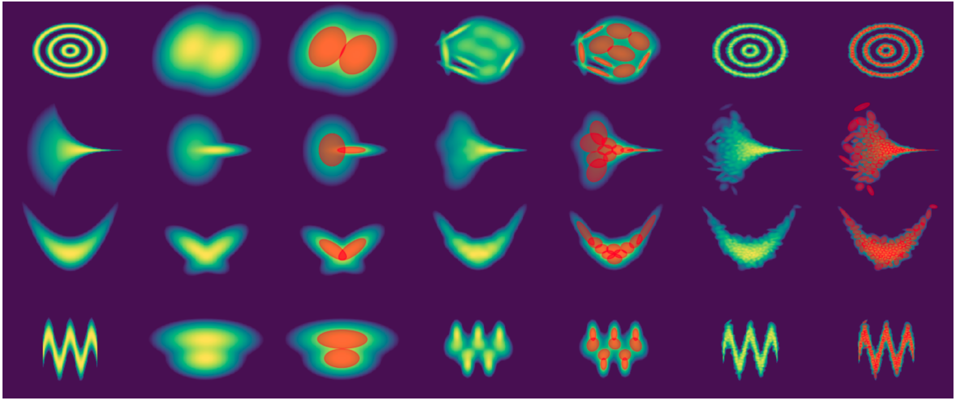
Expressive efficiency (succinctness) compares model sizes in terms of their ability to compactly represent functions

$\Longrightarrow$ *but how many components do they 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

Mixture models

Expressive efficiency



⇒ *deeper mixtures would be efficient compared to shallow ones*

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?*

pinterest.com/pin/190417890473268205/

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?*

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

pinterest.com/pin/190417890473268205/

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?*

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

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

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

pinterest.com/pin/190417890473268205/

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!

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

pinterest.com/pin/190417890473268205/

Marginal MAP (MMAP)

aka Bayesian Network MAP

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

pinterest.com/pin/190417890473268205/

Marginal MAP (MMAP)

aka Bayesian Network MAP

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

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

pinterest.com/pin/190417890473268205/

Marginal MAP (MMAP)

aka Bayesian Network MAP

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

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

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

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

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

pinterest.com/pin/190417890473268205/

Marginal MAP (MMAP)

aka Bayesian Network MAP

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

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

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

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

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

pinterest.com/pin/190417890473268205/

Advanced queries

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

pinterest.com/pin/190417890473268205/

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?*

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

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

pinterest.com/pin/190417890473268205/

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 Jaffa than Marina?*

pinterest.com/pin/190417890473268205/

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 Jaffa than Marina?*

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

pinterest.com/pin/190417890473268205/

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 Jaffa than Marina?*

and more:

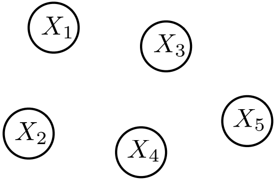
- expected classification agreement *[Oztok et al. 2016; Choi et al. 2017, 2018]*
- expected predictions *[Khosravi et al. 2019a]*

pinterest.com/pin/190417890473268205/

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

Fully factorized models

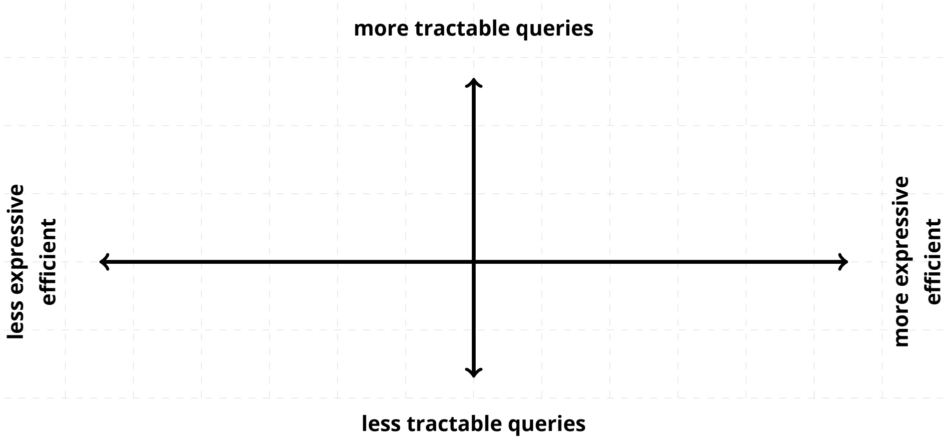
A completely disconnected graph. Example: Product of Bernoullis (PoBs)

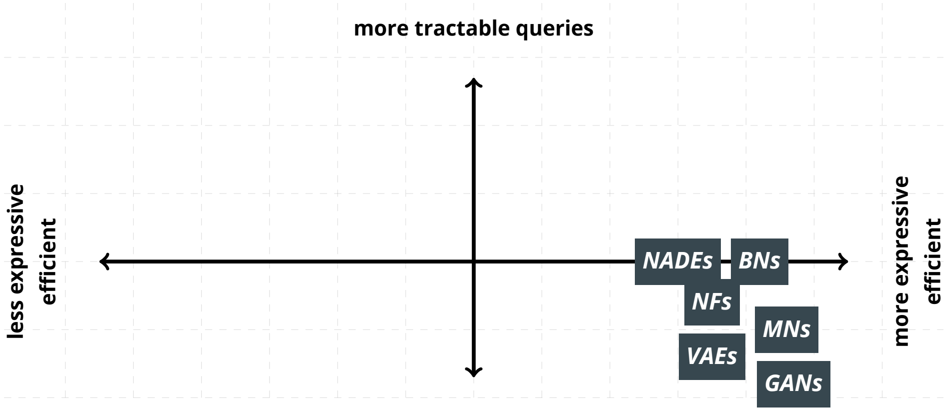


$$p(\mathbf{X}) = \prod_{i=1}^{n} p(x_i)$$

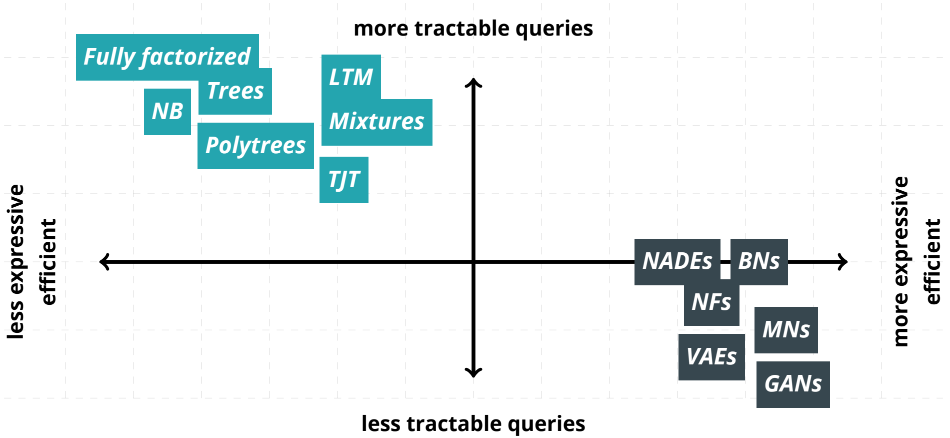
Complete evidence, marginals and MAP, MMAP inference is ***linear***!

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

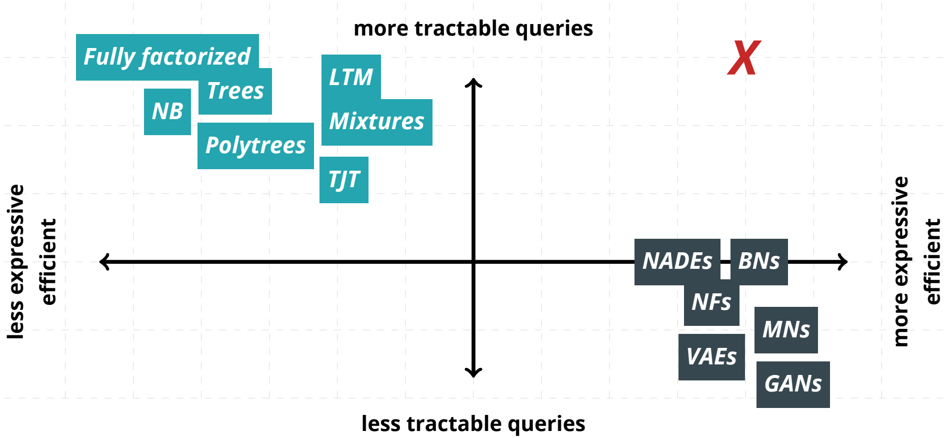

more tractable queries
less expressive
efficient
more expressive
efficient
less tractable queries



Expressive models are not very tractable...



and tractable ones are not very expressive...



probabilistic circuits are at the "sweet spot"

Probabilistic Circuits

Stay Tuned For ...

Next:

1. *What are the building blocks of tractable models?*

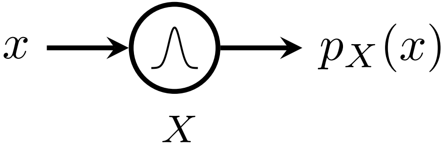
   $\Rightarrow$ *a computational graph forming a probabilistic circuit*

2. For which queries are probabilistic circuits tractable?

   $\Rightarrow$ *tractable classes induced by structural properties*

After: *How are probabilistic circuits related to the alphabet soup of models?*

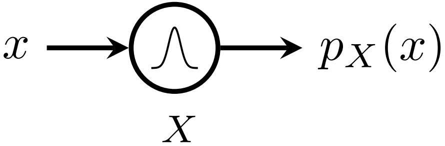
Base Case: Univariate Distributions



Generally, univariate distributions are tractable for:

- EVI: output $p(X_i)$ (density or mass)
- MAR: output 1 (normalized) or Z (unnormalized)
- MAP: output the mode

Base Case: Univariate Distributions



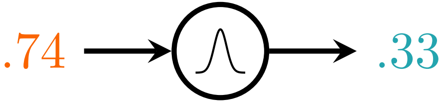
Generally, univariate distributions are tractable for:

- EVI: output $p(X_i)$ (density or mass)
- MAR: output 1 (normalized) or Z (unnormalized)
- MAP: output the mode

$\Longrightarrow$ *often 100% probability for one value of a categorical random variable*

$\Longrightarrow$ *for example, X or $\neg X$ for Boolean random variable*

Base Case: Univariate Distributions



X

Generally, univariate distributions are tractable for:

- EVI: output $p(X_i)$ (density or mass)
- MAR: output 1 (normalized) or Z (unnormalized)
- MAP: output the mode

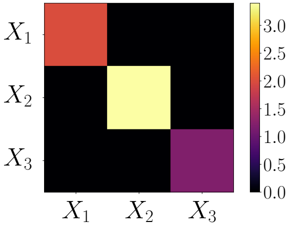
$\Longrightarrow$ *often 100% probability for one value of a categorical random variable*

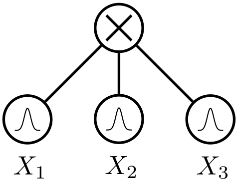
$\Longrightarrow$ *for example, X or $\neg X$ for Boolean random variable*

Factorizations are products

Divide and conquer complexity

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



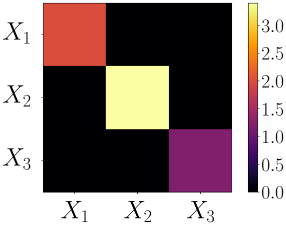


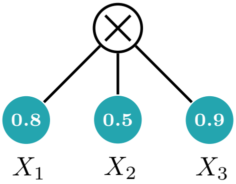
$\Longrightarrow$ *e.g. modeling a multivariate Gaussian with diagonal covariance matrix*

Factorizations are products

Divide and conquer complexity

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



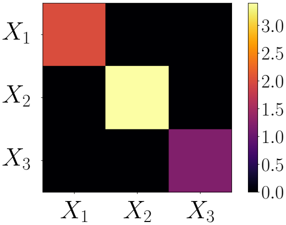


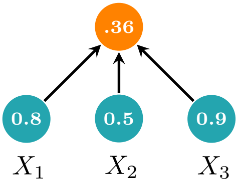
$\Longrightarrow$ *e.g. modeling a multivariate Gaussian with diagonal covariance matrix*

Factorizations are products

Divide and conquer complexity

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

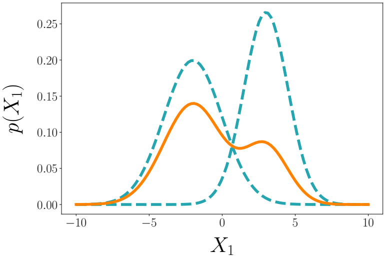




$\Longrightarrow$ *e.g. modeling a multivariate Gaussian with diagonal covariance matrix*

Mixtures are sums

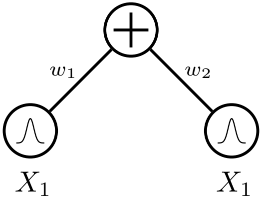
Also mixture models can be treated as a simple ***computational unit*** over distributions



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

Mixtures are sums

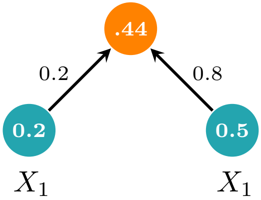
Also mixture models can be treated as a simple ***computational unit*** over distributions



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

Mixtures are sums

Also mixture models can be treated as a simple ***computational unit*** over distributions



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

With mixtures, we increase expressiveness

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

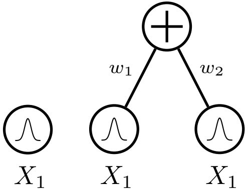
A grammar for tractable models

Recursive semantics of probabilistic circuits

X_1

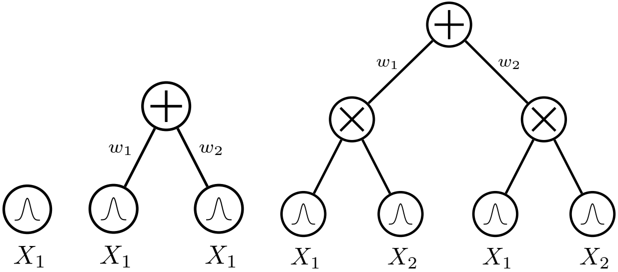
A grammar for tractable models

Recursive semantics of probabilistic circuits



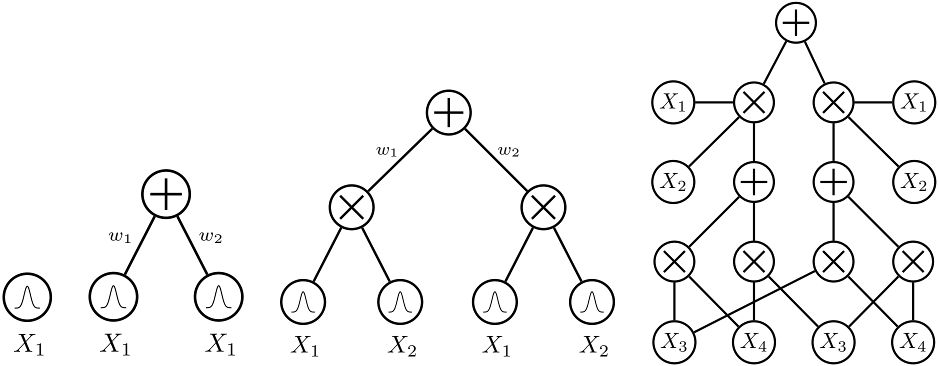
A grammar for tractable models

Recursive semantics of probabilistic circuits



A grammar for tractable models

Recursive semantics of probabilistic circuits



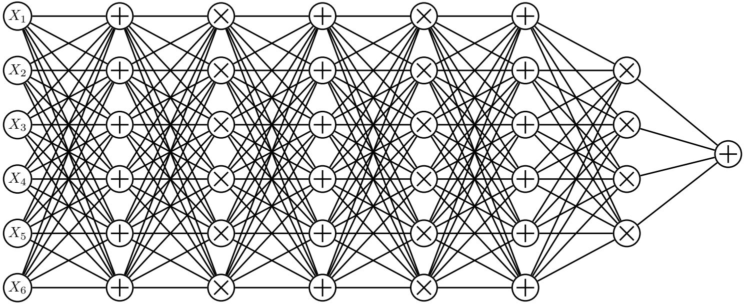
Probabilistic circuits are not PGMs!

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

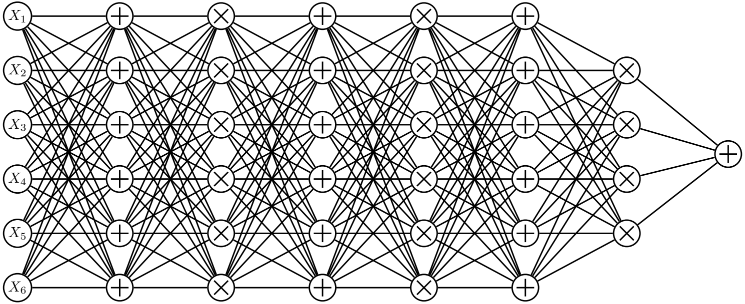
$\Longrightarrow$ *they are computational graphs, more like neural networks*

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!*~~

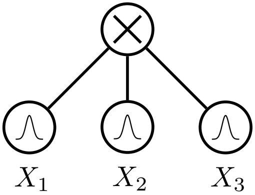
$\Longrightarrow$ structural constraints needed for tractability

How do we ensure tractability?

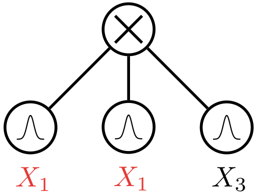
Decomposability

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

$\Rightarrow$ *just like in factorization!*



decomposable circuit



non-decomposable circuit

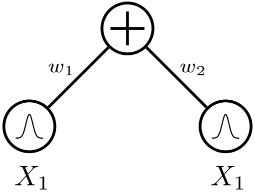
Darwiche et al., "A knowledge compilation map", 2002

Smoothness

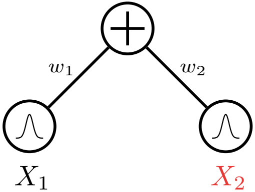
aka completeness

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

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



smooth circuit



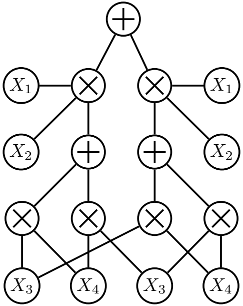
non-smooth circuit

$\Longrightarrow$ *smoothness can be easily enforced [Shih et al. 2019]*

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

Tractable MAR/CON

Smoothness and decomposability enable tractable MAR/CON queries



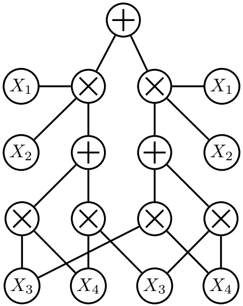
Tractable MAR/CON

Smoothness and decomposability enable tractable MAR/CON queries

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

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

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



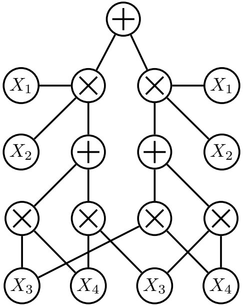
Tractable MAR/CON

Smoothness and decomposability enable tractable MAR/CON queries

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*

Tractable MAR/CON

Smoothness and decomposability enable tractable MAR/CON queries

Forward pass evaluation for MAR

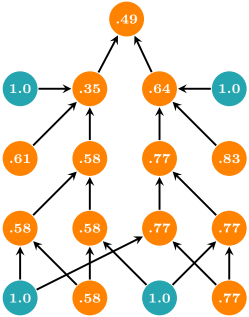
$\Longrightarrow$ *linear in circuit size!*

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

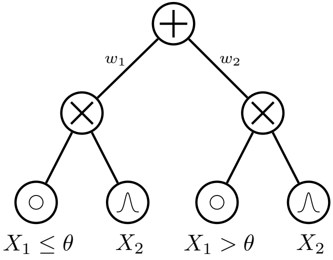


Determinism

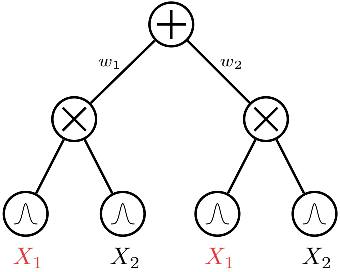
aka selectivity

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

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



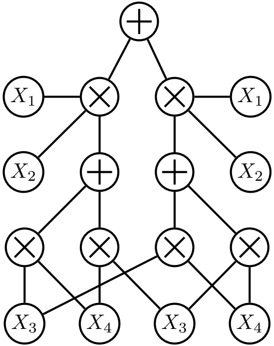
deterministic circuit



non-deterministic circuit

Tractable MAP

The addition of determinism enables tractable MAP queries!



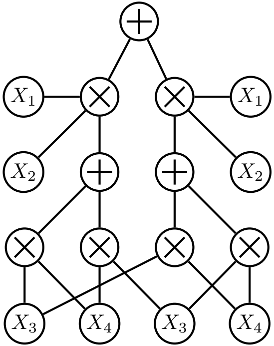
Tractable MAP

The addition of determinism enables tractable MAP queries!

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):

$$\begin{aligned}
\operatorname*{argmax}_{\mathbf{q}} p(\mathbf{q} \mid \mathbf{e}) &= \operatorname*{argmax}_{\mathbf{q}} p(\mathbf{q}, \mathbf{e}) \\
&= \operatorname*{argmax}_{\mathbf{q_x}, \mathbf{q_y}} p(\mathbf{q_x}, \mathbf{e_x}, \mathbf{q_y}, \mathbf{e_y}) \\
&= \operatorname*{argmax}_{\mathbf{q_x}} p(\mathbf{q_x}, \mathbf{e_x}), \operatorname*{argmax}_{\mathbf{q_y}} p(\mathbf{q_y}, \mathbf{e_y})
\end{aligned}$$

$\Longrightarrow$ *solving optimization independently*



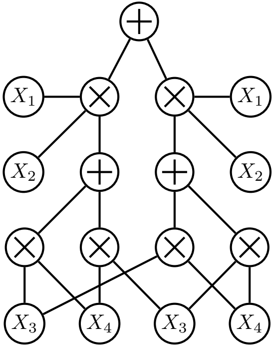
Tractable MAP

The addition of determinism enables tractable MAP queries!

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):

$$\begin{aligned}\operatorname*{argmax}_{\mathbf{q}} p(\mathbf{q},\mathbf{e}) &= \operatorname*{argmax}_{\mathbf{q}} \sum_i w_i p_i(\mathbf{q},\mathbf{e}) \\ &= \operatorname*{argmax}_{\mathbf{q}} \max_i w_i p_i(\mathbf{q},\mathbf{e}) \\ &= \max_i \operatorname*{argmax}_{\mathbf{q}} w_i p_i(\mathbf{q},\mathbf{e})\end{aligned}$$

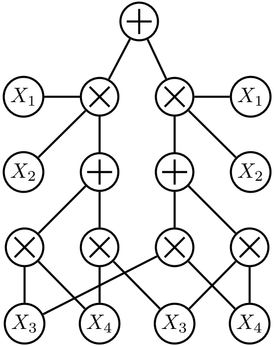
$\Rightarrow$ *one non-zero child term, thus sum is max*



Tractable MAP

The addition of determinism enables tractable MAP queries!

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



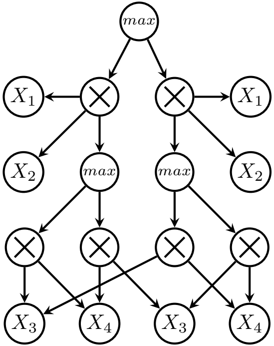
Tractable MAP

The addition of determinism enables tractable MAP queries!

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

In practice:

1. turn sum into max nodes
2. evaluate $p(\mathbf{e})$ bottom-up
3. retrieve max activations top-down
4. compute MAP queries at leaves



Tractable MAP

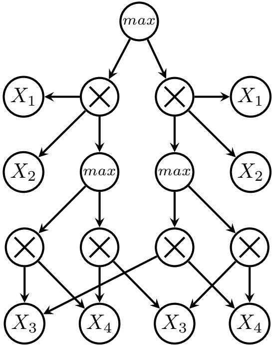
The addition of determinism enables tractable MAP queries!

Evaluating the circuit twice:

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

In practice:

1. turn sum into max nodes
2. evaluate $p(\mathbf{e})$ bottom-up
3. retrieve max activations top-down
4. compute MAP queries at leaves



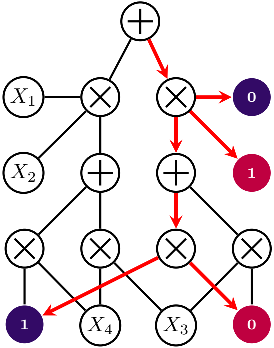
Tractable MAP

The addition of determinism enables tractable MAP queries!

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

In practice:

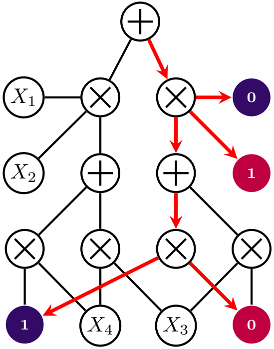
1. turn sum into max nodes
2. evaluate $p(\mathbf{e})$ bottom-up
3. retrieve max activations top-down
4. compute MAP queries at leaves



Tractable MAP

The addition of determinism enables tractable MAP queries!

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



In practice:

1. turn sum into max nodes
2. evaluate $p(\mathbf{e})$ bottom-up
3. retrieve max activations top-down
4. compute MAP queries at leaves

Approximate MAP

If the probabilistic circuit is ***non-deterministic***, MAP is intractable:

$\Rightarrow$ *e.g. with latent variables* $\mathbf{Z}$

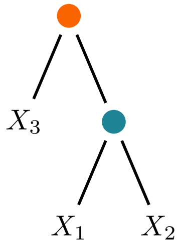
$$\operatorname*{argmax}_{\mathbf{q}} \sum_{i} w_i p_i(\mathbf{q}, \mathbf{e}) = \operatorname*{argmax}_{\mathbf{q}} \sum_{\mathbf{z}} p(\mathbf{q}, \mathbf{z}, \mathbf{e}) \neq \operatorname*{argmax}_{\mathbf{q}} \max_{\mathbf{z}} p(\mathbf{q}, \mathbf{z}, \mathbf{e})$$

However, same two steps algorithm, still used as an approximation to MAP *[Liu et al. 2013; Peharz et al. 2016]*

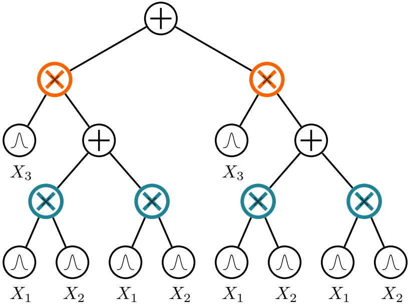
Structured decomposability

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 enables tractable ...

- ***Entropy*** of probabilistic circuit *[Liang et al. 2017b]*
- ***Symmetric*** and ***group queries*** (exactly-k, odd-number, more, 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. 2019b]*

Structured decomposability enables tractable ...

- ***Entropy*** of probabilistic circuit *[Liang et al. 2017b]*
- ***Symmetric*** and ***group queries*** (exactly-k, odd-number, more, 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. 2019b]*

Stay Tuned For ...

Next:

1. *How probabilistic circuits are related to logical ones?*

   $\Longrightarrow$ *a historical perspective*

2. *How probabilistic circuits in the literature relate and differ?*

   $\Longrightarrow$ *SPNs, ACs, CNets, PSDDs*

3. *How classical tractable models can be turned in a circuit?*

   $\Longrightarrow$ *Compiling low-treewidth PGMs*

After: *How do I build my own probabilistic circuit?*

Tractability to other semi-rings

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

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

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

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

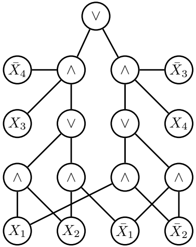
$\Longrightarrow$ *Algebraic model counting [Kimmig et al. 2017], Semi-ring programming [Belle et al. 2016]*

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

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

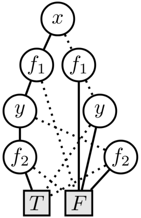
$\Longrightarrow$ *logical circuits!*

Logical circuits



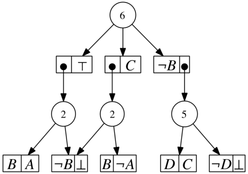
s/d-D/NNFs

[Darwiche et al. 2002]



O/BDDs

[Bryant 1986]



SDDs

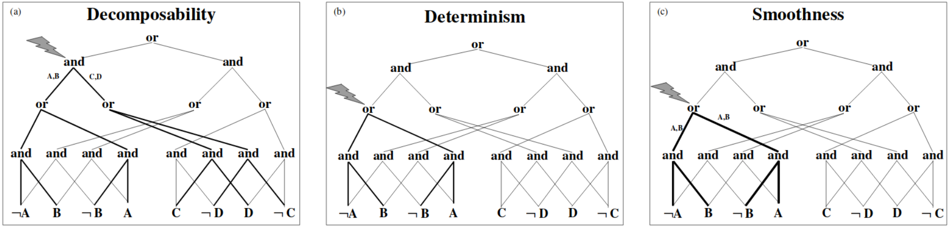
[Darwiche 2011]

Logical circuits are compact representations for boolean functions...

Logical circuits

structural properties

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

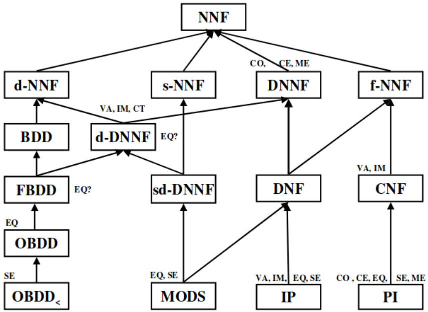


Darwiche et al., “A knowledge compilation map”, 2002

Logical circuits

a knowledge compilation map

...inducing ***a hierarchy of tractable query classes***



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

Logical circuits

connection to probabilistic circuits through WMC

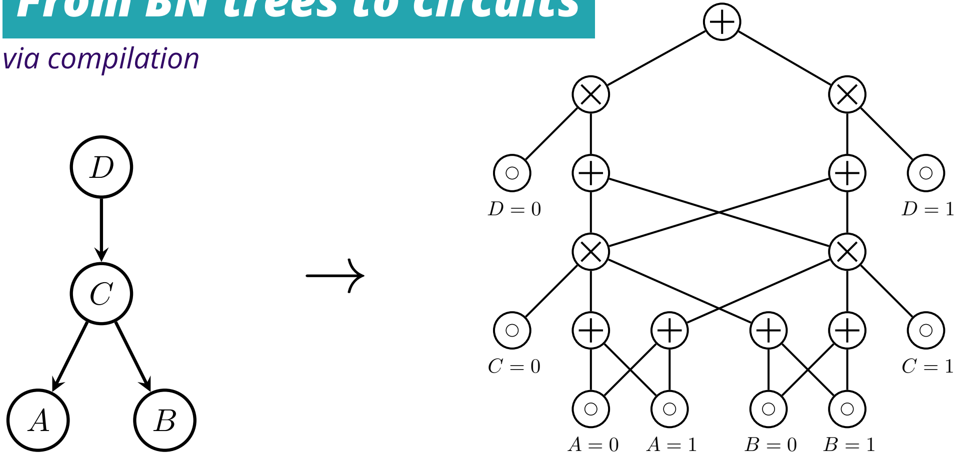
- A task called ***weighted model counting*** (**WMC**)

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

- Two decades worth of connections:
  1. Encode probabilistic model as WMC (add variable placeholders for parameters)
  2. Compile Δ into a d-DNNF (or OBDD, SDD, etc.)
  3. Tractable MAR/CON by tractable WMC on circuit
  4. Depending on the WMC encoding even tractable MAP
- End result equivalent to probabilistic circuit: efficiently replace parameter variables in logical circuit by edge parameters in probabilistic circuit

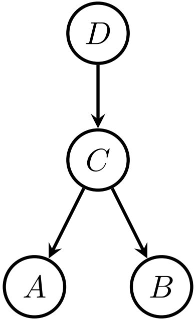
From BN trees to circuits

via compilation



From BN trees to circuits

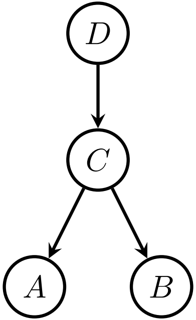
via compilation



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

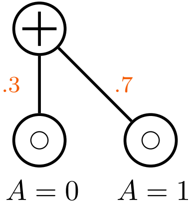
From BN trees to circuits

via compilation



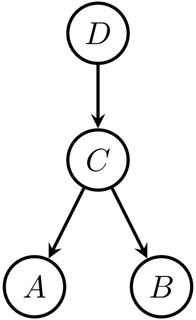
...compile a leaf CPT

$p(A|C=0)$

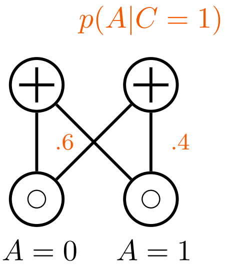


From BN trees to circuits

via compilation

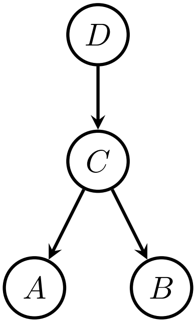


...compile a leaf CPT

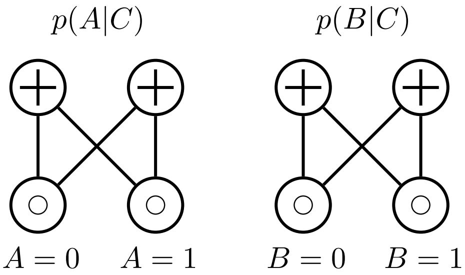


From BN trees to circuits

via compilation

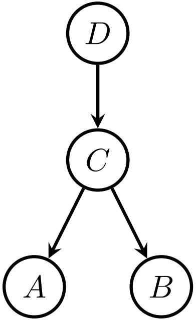


...compile a leaf CPT...for all leaves...

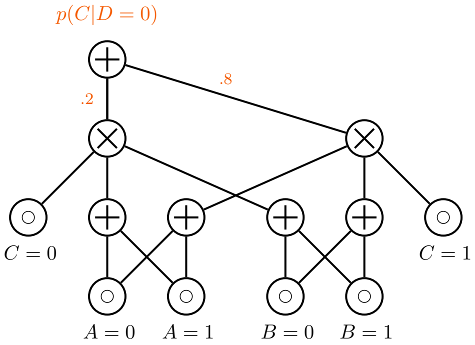


From BN trees to circuits

via compilation



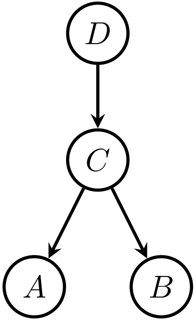
...and recurse over parents...

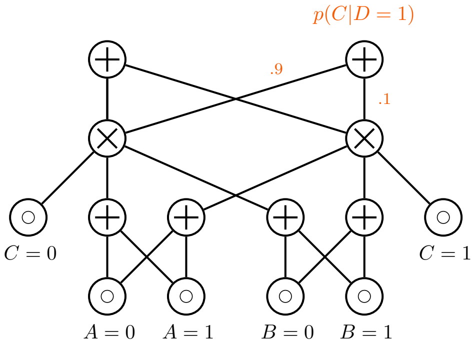


From BN trees to circuits

via compilation

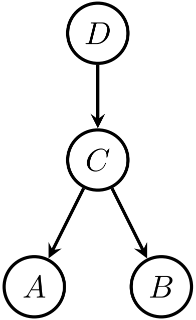
…while reusing previously compiled nodes!…

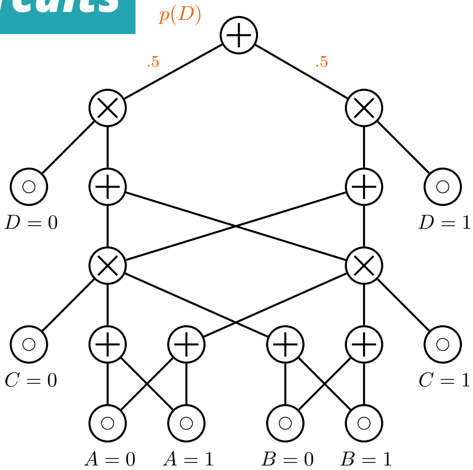




From BN trees to circuits

via compilation





Low-treewidh PGMs

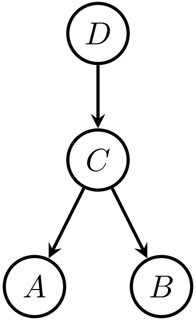
Tree, polytrees and thin junction trees can be turned into

- decomposable
- smooth
- deterministic

probabilistic circuits

Therefore they support tractable

- EVI
- MAR/CON
- MAP



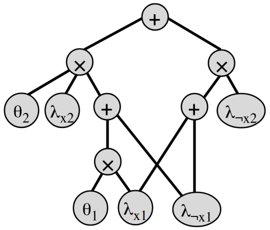
Arithmetic Circuits (ACs)

ACs *[Darwiche 2003]* are

- decomposable
- smooth
- deterministic

They support tractable

- EVI
- MAR/CON
- (MAP)



$\Rightarrow$ *parameters attached to leaves (cf. WMC) ...can be moved to sum nodes*

$\Rightarrow$ *Support for tractable MAP queries depends on intended WMC encoding*

$\Rightarrow$ *Also see related AND/OR search spaces [Dechter et al. 2007]*

Lowd et al., "Learning Markov Networks With Arithmetic Circuits", 2013

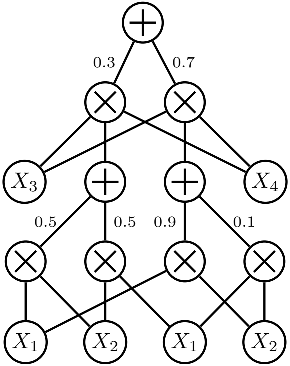
Sum-Product Networks (SPNs)

SPNs *[Poon et al. 2011]* are

- decomposable
- smooth
- ~~deterministic~~

They support tractable

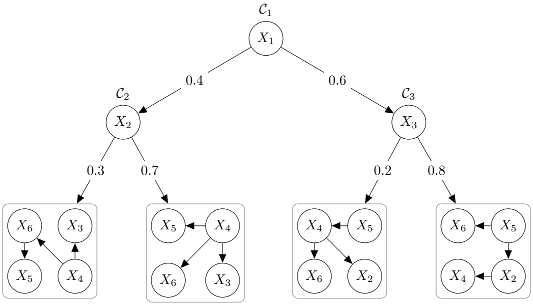
- EVI
- MAR/CON
- ~~MAP~~



$\Longrightarrow$ *deterministic SPNs are also called* selective *[Peharz et al. 2014]*

Cutset Networks (CNets)

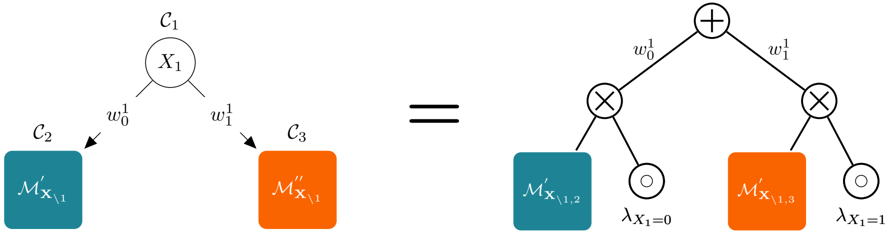
A CNet *[Rahman et al. 2014]* is a ***weighted model tree*** *[Dechter et al. 2007]* whose leaves are tree Bayesian networks



$\Longrightarrow$ *they can be represented as probabilistic circuits*

CNets as probabilistic circuits

Every ***decision node*** in the CNet can be represented as a deterministic, smooth sum node



and we can recurse on each child node until a BN tree is reached

$\Longrightarrow$ *compilable into a deterministic, smooth and decomposable circuit!*

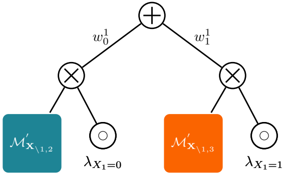
CNets as probabilistic circuits

CNets are

- decomposable
- smooth
- deterministic

They support tractable

- EVI
- MAR/CON
- MAP



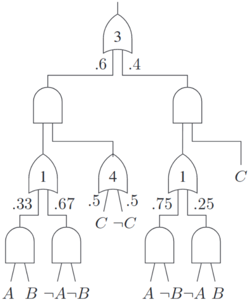
Probabilistic Sentential Decision Diagrams

PSDDs *[Kisa et al. 2014a]* are

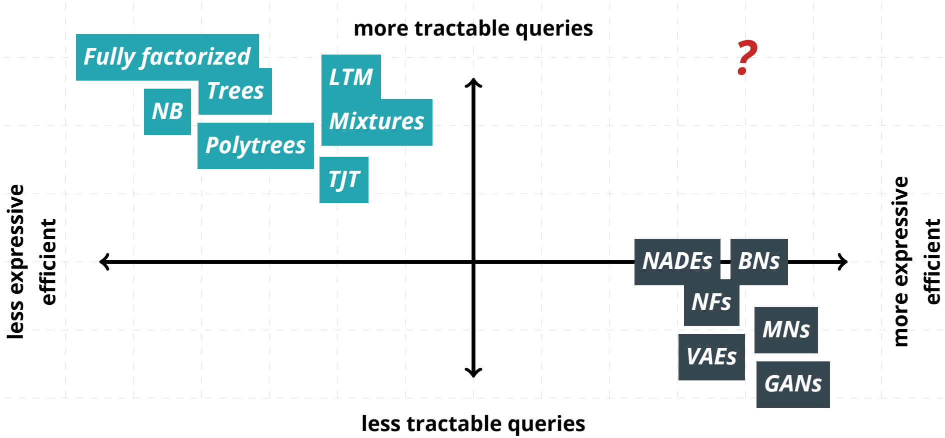
- structured decomposable
- smooth
- deterministic

They support tractable

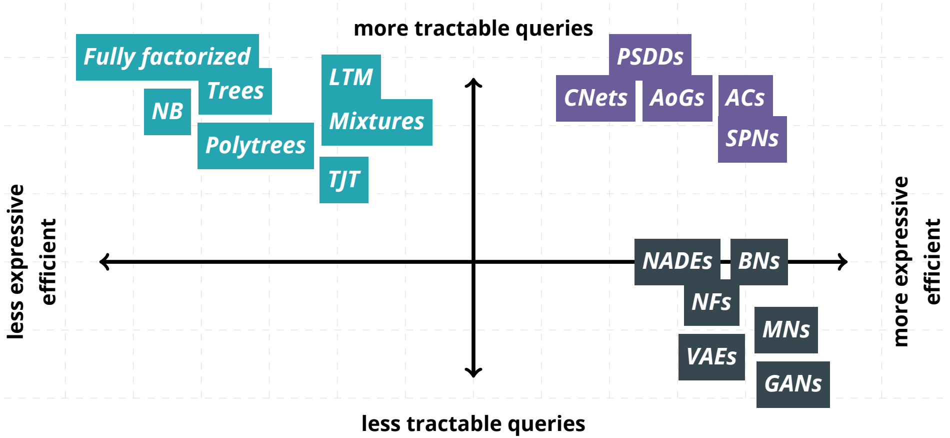
- EVI
- MAR/CON
- MAP
- Complex queries!



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



where are probabilistic circuits?



tractability vs expressive efficiency

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
Peharz et al., "Probabilistic deep learning using random sum-product networks", 2018

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

Read more in online slides about ...

Building Circuits:

1. *How to learn circuit parameters?*

   $\Longrightarrow$ *convex optimization, EM, SGD, Bayesian learning, ...*

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

   $\Longrightarrow$ *local search, random structures, ensembles, ...*

3. *How to compile other models to circuits?*

   $\Longrightarrow$ *PGM compilation, probabilistic databases, probabilistic programming*

See: `http://starai.cs.ucla.edu/slides/TPMTutorialUAI19.pdf`

Tractable Learning

A learner L *is a* tractable learner *for a class of queries* $\mathcal{Q}$ *iff*

(1) for any dataset $\mathcal{D}$*, learner* $L(\mathcal{D})$ *runs in time* $O(\text{poly}(|\mathcal{D}|))$*, and*

(2) outputs a probabilistic model that is tractable for queries $\mathcal{Q}$.

Tractable Learning

A learner L *is a* tractable learner *for a class of queries* $\mathcal{Q}$ *iff*

(1) for any dataset $\mathcal{D}$*, learner* $L(\mathcal{D})$ *runs in time* $O(\text{poly}(|\mathcal{D}|))$*, and*

$\implies$ *Guarantees learned model has size* $O(\text{poly}(|\mathcal{D}|))$

$\implies$ *Guarantees learned model has size* $O(\text{poly}(|\mathbf{X}|))$

(2) outputs a probabilistic model that is tractable for queries $\mathcal{Q}$.

Tractable Learning

A learner L *is a* tractable learner *for a class of queries* $\mathcal{Q}$ *iff*

(1) for any dataset $\mathcal{D}$, *learner* $L(\mathcal{D})$ *runs in time* $O(\text{poly}(|\mathcal{D}|))$, *and*

$\implies$ *Guarantees learned model has size* $O(\text{poly}(|\mathcal{D}|))$

$\implies$ *Guarantees learned model has size* $O(\text{poly}(|\mathbf{X}|))$

(2) outputs a probabilistic model that is tractable for queries $\mathcal{Q}$.

$\implies$ *Guarantees efficient querying for* $\mathcal{Q}$ *in time* $O(\text{poly}(|\mathbf{X}|))$

Stay Tuned For ...

Next:

1. *How to learn circuit parameters?*

   $\Rightarrow$ *convex optimization, EM, SGD, Bayesian learning, ...*

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

   $\Rightarrow$ *local search, random structures, ensembles, ...*

After: *What is this used for?*

Learning circuit parameters

The parameters of a probabilistic circuit are

sum node parameters $\boldsymbol{w}$ + input distributions' parameters $\boldsymbol{\theta}$

$\Longrightarrow$ *e.g., parameters of Bernoulli or Gaussian leaves*

Recall that if a sum node is **non-deterministic**, it marginalizes out latent variables Z...

$\Longrightarrow$ *i.e., we are training a mixture model*

Learning circuit parameters

deterministic circuits $\Rightarrow$ closed-form, convex optimization *[Kisa et al. 2014b; Liang et al. 2019]*

non- deterministic circuits $\Rightarrow$

- SGD *[Peharz et al. 2018]*
- soft/hard EM *[Poon et al. 2011; Peharz 2015]*
- bayesian moment matching *[Jaini et al. 2016]*
- collapsed variational Bayes *[Zhao et al. 2016a]*
- CCCP *[Zhao et al. 2016b]*
- Extended Baum-Welch *[Rashwan et al. 2018]*

Deterministic circuits

Given a deterministic circuit $\mathcal{C}$ and a complete dataset $\mathbf{D}$, the likelihood of $\mathcal{C}$ given $\mathbf{D}$ is

$$L(\boldsymbol{w}; \mathbf{D}) = \prod_{\mathbf{x} \in \mathbf{D}} p_{\mathcal{C}}(\mathbf{x}; \boldsymbol{w})$$

as it decomposes as in BNs, the MLE parameters are computable as

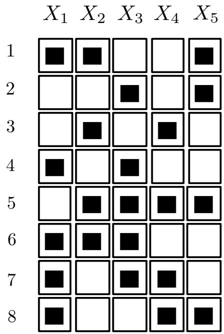
$$w_{i,j}^{\mathsf{MLE}} = \frac{\sum_{d \in \mathbf{D}} \mathbb{1}\{\mathbf{x} \models [i \wedge j]\}}{\sum_{d \in \mathbf{D}} \mathbb{1}\{\mathbf{x} \models [i]\}}$$

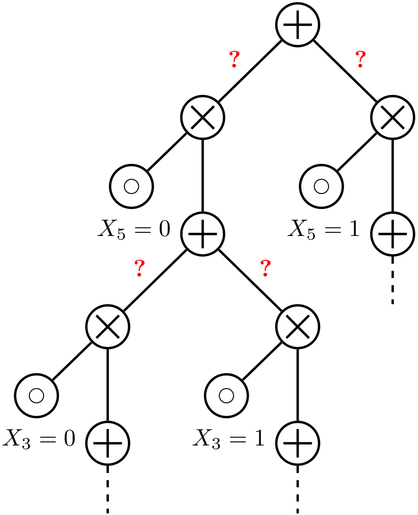
$\implies$ *compute sufficient statistics (just count) in a single pass of* $\mathbf{D}$

Kisa et al., "Probabilistic sentential decision diagrams", 2014
Liang et al., "Learning Logistic Circuits", 2019

Deterministic circuits

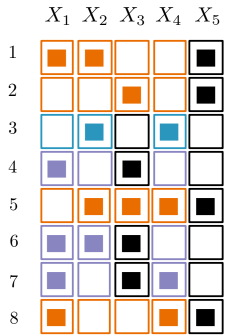
An example

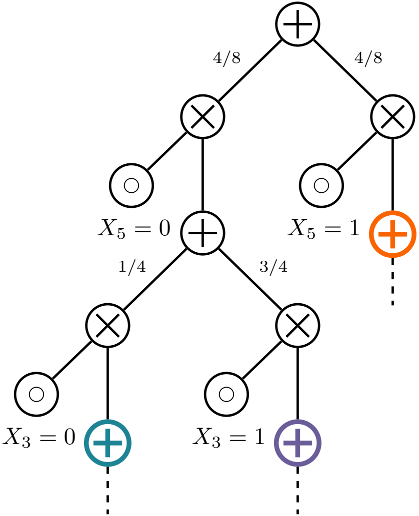




Deterministic circuits

An example





Non-deterministic circuits

Gradient Descent

Computing the likelihood gradient and optimize by GD

	Δw_{pc}
Soft Gradient	
Generative ($\nabla_{w_{pc}} S(\mathbf{x})$)	$S_c(\mathbf{x}) \nabla_{S_p(\mathbf{x})} S(\mathbf{x})$
Discriminative ($\nabla_{w_{pc}} \log S(\mathbf{y}\|\mathbf{x})$)	$\frac{\nabla_{w_{pc}} S(\mathbf{y}\|\mathbf{x})}{S(\mathbf{y}\|\mathbf{x})} - \frac{\nabla_{w_{pc}} S(*\|\mathbf{x})}{S(*\|\mathbf{x})}$
Hard Gradient	
Generative ($\nabla_{w_{pc}} \log M(\mathbf{x})$)	$\frac{\sharp\{w_{pc} \in W_{\mathbf{x}}\}}{w_{pc}}$
Discriminative ($\nabla_{w_{pc}} \log M(\mathbf{y}\|\mathbf{x})$)	$\frac{\sharp\{w_{pc} \in W_{(\mathbf{y}\|\mathbf{x})}\} - \sharp\{w_{pc} \in W_{(\mathbf{1}\|\mathbf{x})}\}}{w_{pc}}$

Gens et al., "Discriminative Learning of Sum-Product Networks", 2012

Non-deterministic circuits

Expectation Maximization

...or using EM by considering each sum node as the marginalization of a hidden variable

Soft Posterior $(p(H_p = c\|\mathbf{x}))$	$\propto$	$\frac{1}{S(\mathbf{x})} \frac{\partial S(\mathbf{x})}{\partial S_p(\mathbf{x})} S_c(\mathbf{x}) w_{pc}$
Hard Posterior $(p(H_p = c\|\mathbf{x}))$	$=$	$\begin{cases} 1 \text{ if } w_{pc} \in W_{\mathbf{x}} \\ 0 \text{ otherwise} \end{cases}$

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

Bayesian Parameter Learning

Bayesian Learning starts by expressing a prior $p(\mathbf{w})$ over the weights

$\Longrightarrow$ *learning corresponds to computing the posterior based on the data*

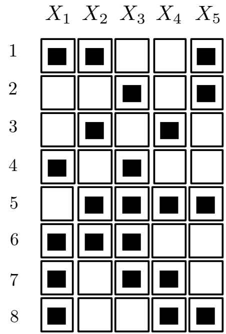
$$p(\mathbf{w}|\mathcal{D}) \propto p(\mathbf{w})p(\mathcal{D}|\mathbf{w})$$

- Moment matching (oBMM) *[Rashwan et al. 2016]*
- oBMM extended with Gaussian distributions *[Jaini et al. 2016]*
- collapsed variational inference algorithm *[Zhao et al. 2016b]*

Structure learning

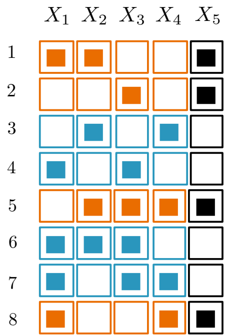
- ***greedy top-down***: LearnSPN and variants
- ***hill climbing***: LearnPSDD and variants
- ***random structures***: RAT-SPNs, XCNets, ...
- ***ensembles*** of circuits: EM, bagging, boosting,...

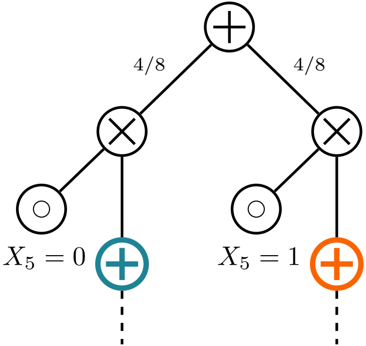
LearnCNet



Di Mauro et al., "Learning Accurate Cutset Networks by Exploiting Decomposability", 2015

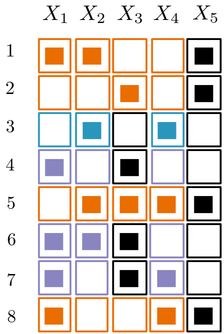
LearnCNet

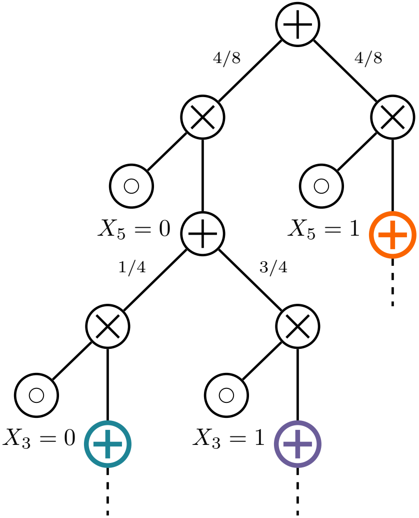




Di Mauro et al., "Learning Accurate Cutset Networks by Exploiting Decomposability", 2015

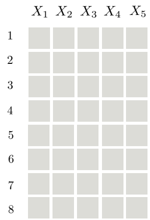
LearnCNet





Di Mauro et al., "Learning Accurate Cutset Networks by Exploiting Decomposability", 2015

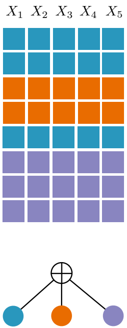
LearnSPN



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

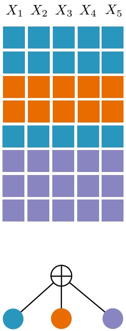
LearnSPN

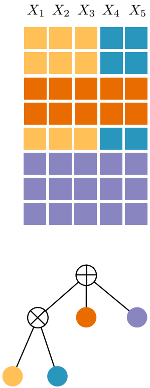


Looking for sub-population in the data—***clustering***—to introduce sum nodes…

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

LearnSPN

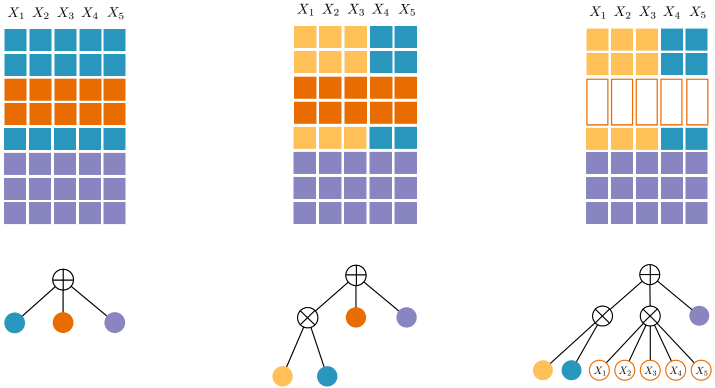




...***seeking independencies among sets of RVs*** to factorize into product nodes

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

LearnSPN



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

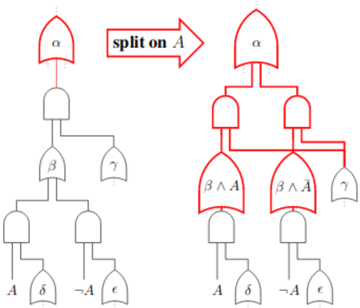
Gens et al., "Learning the Structure of Sum-Product Networks", 2013

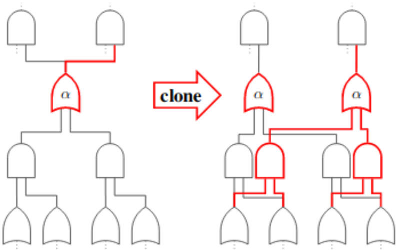
LearnSPN variants

- **ID-SPN** *[Rooshenas et al. 2014]*
- **LearnSPN-b/T/B** *[Vergari et al. 2015]*
- for **heterogeneous data** *[Bueff et al. 2018; Molina et al. 2018]*
- using **k-means** *[Butz et al. 2018]* or **SVD** splits *[Adel et al. 2015]*
- learning **DAGs** *[Dennis et al. 2015; Jaini et al. 2018]*
- **approximating** independence tests *[Di Mauro et al. 2018]*

LearnPSDD

Vtree learning + hill climbing





- local search (split /clone) to maximise a penalized likelihood score

---

Liang et al., "Learning the structure of probabilistic sentential decision diagrams", 2017

Randomized structure learning

Random Tensorized SPNs (RAT-SPNs) *[Peharz et al. 2018]*

- SPNs are obtained by first constructing a random region graph
- subsequently populating the region graph with tensors of SPN nodes
- discriminative+generative parameter learning (SGD/EM + dropout)

Extremely Randomized CNets (XCNets) *[Di Mauro et al. 2017]*

- top-down random conditioning
- learning Chow-Liu trees at the leaves

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
- bagging
- boosting

Bagging

- more efficient than EM
- mixture coefficients are set equally probable
- mixture components can be learned independently on different ***bootstraps***

Adding **random subspace projection** to bagged networks (like for CNets)

- more efficient than bagging

Di Mauro et al., "Learning Accurate Cutset Networks by Exploiting Decomposability", 2015
Di Mauro et al., "Learning Bayesian Random Cutset Forests", 2015

Boosting

Boosting Probabilistic Circuits

- BDE: boosting density estimation
  - sequentially grows the ensemble, adding a weak base learner at each stage
  - at each boosting step m, find a weak learner c_m and a coefficient η_m maximizing the weighted LL of the new model

$$f_m = (1 - \eta_m) f_{m-1} + \eta_m c_m$$

- GBDE: a kernel based generalization of BDE—AdaBoost style algorithm
- sequential EM
  - at each step m, jointly optimize η_m and c_m keeping f_{m-1} fixed

Rahman et al., "Learning Ensembles of Cutset Networks", 2016

Applications

Read more in online slides about ...

Applications:

1. *How to compile other models to circuits?*

   $\Longrightarrow$ *PGM compilation, probabilistic databases, probabilistic programming*

2. what have probabilistic circuits been used for?

   $\Longrightarrow$ *computer vision, sop, speech, planning, ...*

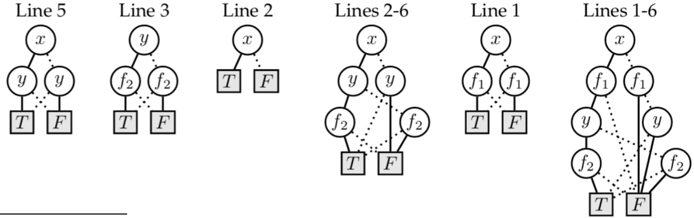
3. what are the current trends and challenges?

   $\Longrightarrow$ *hybrid models, benchmarks, scaling, reasoning*

See: `http://starai.cs.ucla.edu/slides/TPMTutorialUAI19.pdf`

Probabilistic programming

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

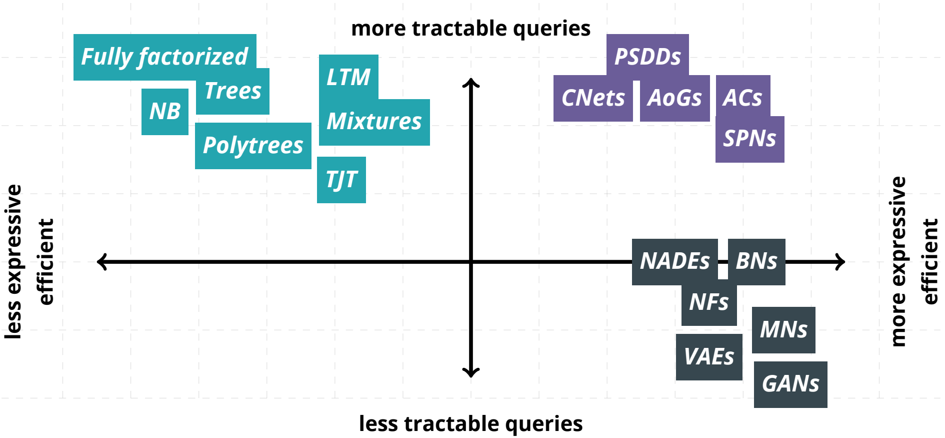


Chavira et al., "Compiling relational Bayesian networks for exact inference", 2006
Holtzen et al., "Symbolic Exact Inference for Discrete Probabilistic Programs", 2019
De Raedt et al.; Riguzzi; Fierens et al.; Vlasselaer et al., "ProbLog: A Probabilistic Prolog and Its Application in Link Discovery."; "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

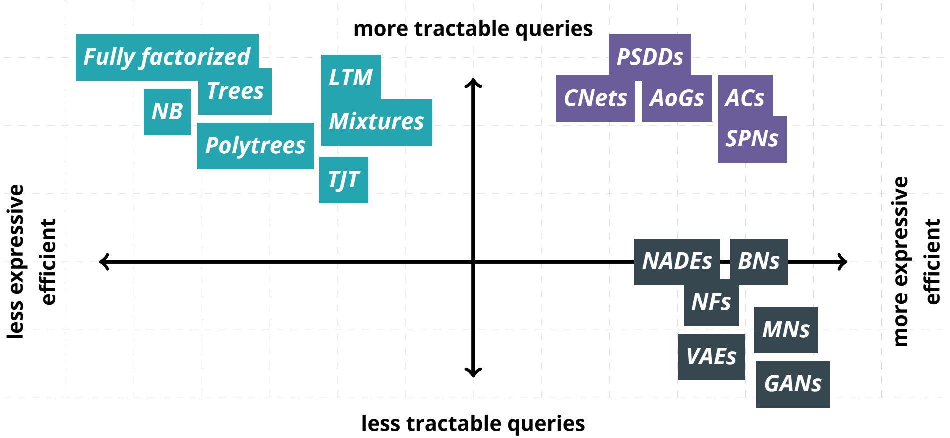
The Logical Conclusions

- Logical roots of probabilistic circuits
- Probabilistic circuits bridge between logic and deep learning
- Bring back world models!
- Powerful general reasoning tool

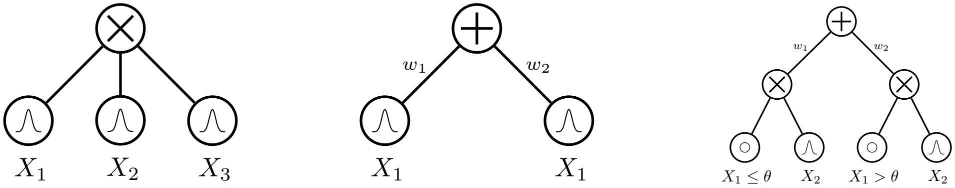
  $\Rightarrow$ *for example in probabilistic programming*
- Elegant knowledge representation formalism



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

Code

We will soon release **Juice** v0.1: *The Julia Circuit Empanada*

- Learning probabilistic circuits from data and choose to be
  - decomposable – deterministic – structured decomposable
- Evaluate tractable queries
  - EVI – MAR, COND – MAP – Complex queries, expectations, etc.
- Easily compile logical and probabilistic circuits from other representations
- Highly efficient using Julia's SIMD processing capabilities

Tractable Probabilistic Models

Representations
Inference
Learning
Applications

Guy Van den Broeck
University of California, Los Angeles

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

Antonio Vergari
University of California, Los Angeles

YooJung Choi
University of California, Los Angeles

Robert Peharz
TU Eindhoven

Nicola Di Mauro
University of Bari

November 11, 2019 - ***International Spring School on "Uncertainty in AI and data management"*** **- Santiago, Chile**

References I

- Chow, C and C Liu (1968). “Approximating discrete probability distributions with dependence trees”. In: *IEEE Transactions on Information Theory* 14.3, pp. 462–467.
- Bryant, R (1986). “Graph-based algorithms for boolean manipulation”. In: *IEEE Transactions on Computers*, pp. 677–691.
- Cooper, Gregory F (1990). “The computational complexity of probabilistic inference using Bayesian belief networks”. In: *Artificial intelligence* 42.2-3, pp. 393–405.
- Dagum, Paul and Michael Luby (1993). “Approximating probabilistic inference in Bayesian belief networks is NP-hard”. In: *Artificial intelligence* 60.1, pp. 141–153.
- Zhang, Nevin Lianwen and David Poole (1994). “A simple approach to Bayesian network computations”. In: *Proceedings of the Biennial Conference-Canadian Society for Computational Studies of Intelligence*, pp. 171–178.
- Roth, Dan (1996). “On the hardness of approximate reasoning”. In: *Artificial Intelligence* 82.1–2, pp. 273–302.
- Dechter, Rina (1998). “Bucket elimination: A unifying framework for probabilistic inference”. In: *Learning in graphical models*. Springer, pp. 75–104.
- Dasgupta, Sanjoy (1999). “Learning polytrees”. In: *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., pp. 134–141.
- 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.

References II

- 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.
- 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.

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.
- Campos, Cassio Polpo de (2011). "New complexity results for MAP in Bayesian networks". In: *IJCAI*. Vol. 11, pp. 2100–2106.
- Darwiche, Adnan (2011). "SDD: A New Canonical Representation of Propositional Knowledge Bases". In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*. IJCAI'11. Barcelona, Catalonia, Spain. ISBN: 978-1-57735-514-4.
- 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.
- Gens, Robert and Pedro Domingos (2012). "Discriminative Learning of Sum-Product Networks". In: *Advances in Neural Information Processing Systems 25*, pp. 3239–3247.
- — (2013). "Learning the Structure of Sum-Product Networks". In: *Proceedings of the ICML 2013*, pp. 873–880.
- Liu, Qiang and Alexander Ihler (2013). "Variational algorithms for marginal MAP". In: *The Journal of Machine Learning Research* 14.1, pp. 3165–3200.

References IV

- Lowd, Daniel and Amirmohammad Rooshenas (2013). “Learning Markov Networks With Arithmetic Circuits”. In: *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics*. Vol. 31. JMLR Workshop Proceedings, pp. 406–414.
- Goodfellow, Ian 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.
- Peharz, Robert, Robert Gens, and Pedro Domingos (2014). “Learning Selective Sum-Product Networks”. In: *Workshop on Learning Tractable Probabilistic Models*. LTPM.
- Rahman, Tahrima, Prasanna Kothalkar, and Vibhav Gogate (2014). “Cutset Networks: A Simple, Tractable, and Scalable Approach for Improving the Accuracy of Chow-Liu Trees”. In: *Machine Learning and Knowledge Discovery in Databases*. Vol. 8725. LNCS. Springer, pp. 630–645.
- Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). “Stochastic backprop. and approximate inference in deep generative models”. In: *arXiv preprint arXiv:1401.4082*.
- Rooshenas, Amirmohammad and Daniel Lowd (2014). “Learning Sum-Product Networks with Direct and Indirect Variable Interactions”. In: *Proceedings of ICML 2014*.

References V

- Adel, Tameem, David Balduzzi, and Ali Ghodsi (2015). “Learning the Structure of Sum-Product Networks via an SVD-based Algorithm”. In: *Uncertainty in Artificial Intelligence*.
- Bekker, Jessa 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)*.
- Dennis, Aaron and Dan Ventura (2015). “Greedy Structure Search for Sum-product Networks”. In: *IJCAI'15*. Buenos Aires, Argentina: AAAI Press, pp. 932–938. ISBN: 978-1-57735-738-4.
- Di Mauro, Nicola, Antonio Vergari, and Floriana Esposito (2015a). “Learning Accurate Cutset Networks by Exploiting Decomposability”. In: *Proceedings of AIXIA*. Springer, pp. 221–232.
- Di Mauro, Nicola, Antonio Vergari, and Teresa M.A. Basile (2015b). “Learning Bayesian Random Cutset Forests”. In: *Proceedings of ISMIS*. Springer, pp. 122–132.
- 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.
- Peharz, Robert (2015). “Foundations of Sum-Product Networks for Probabilistic Modeling”. PhD thesis. Graz University of Technology, SPSC.
- Vergari, Antonio, Nicola Di Mauro, and Floriana Esposito (2015). “Simplifying, Regularizing and Strengthening Sum-Product Network Structure Learning”. In: *ECML-PKDD 2015*.

References VI

- 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`.
- Belle, Vaishak and Luc De Raedt (2016). "Semiring Programming: A Framework for Search, Inference and Learning". In: *arXiv preprint arXiv:1609.06954*.
- 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.
- 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`.
- 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`.
- 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.
- 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.

References VII

- 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.
- 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)*.
- Di Mauro, Nicola et al. (2017). “Fast and Accurate Density Estimation with Extremely Randomized Cutset Networks”. In: *ECML-PKDD 2017*.
- Kimmig, Angelika, Guy Van den Broeck, and Luc De Raedt (2017). “Algebraic model counting”. In: *Journal of Applied Logic* 22, pp. 46–62.
- 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 VIII

- 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.
- Bueff, Andreas, Stefanie Speichert, and Vaishak Belle (2018). “Tractable Querying and Learning in Hybrid Domains via Sum-Product Networks”. In: *arXiv preprint arXiv:1807.05464*.
- Butz, Cory J et al. (2018). “An Empirical Study of Methods for SPN Learning and Inference”. In: *International Conference on Probabilistic Graphical Models*, pp. 49–60.
- Choi, YooJung and Guy Van den Broeck (2018). “On robust trimming of Bayesian network classifiers”. In: *arXiv preprint arXiv:1805.11243*.
- Di Mauro, Nicola et al. (2018). “Sum-Product Network structure learning by efficient product nodes discovery”. In: *Intelligenza Artificiale* 12.2, pp. 143–159.
- 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.
- Jaini, Priyank, Amur Ghose, and Pascal Poupart (2018). “Prometheus: Directly Learning Acyclic Directed Graph Structures for Sum-Product Networks”. In: *International Conference on Probabilistic Graphical Models*, pp. 181–192.
- Molina, Alejandro et al. (2018). “Mixed Sum-Product Networks: A Deep Architecture for Hybrid Domains”. In: *AAAI*.
- Peharz, Robert et al. (2018). “Probabilistic deep learning using random sum-product networks”. In: *arXiv preprint arXiv:1806.01910*.
- 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.

References IX

- Shen, Yujia, Arthur Choi, and Adnan Darwiche (2018). "Conditional PSDDs: Modeling and learning with modular knowledge". In: *Thirty-Second AAAI Conference on Artificial Intelligence*.
- Dai, Bin and David Wipf (2019). "Diagnosing and enhancing vae models". In: *arXiv preprint arXiv:1903.05789*.
- 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). "What to Expect of Classifiers? Reasoning about Logistic Regression with Missing Features". In: *arXiv preprint arXiv:1903.01620*.
- Khosravi, Pasha et al. (2019b). "What to Expect of Classifiers? Reasoning about Logistic Regression with Missing Features". In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*.
- Liang, Yitao and Guy Van den Broeck (2019). "Learning Logistic Circuits". In: *Proceedings of the 33rd Conference on Artificial Intelligence (AAAI)*.
- Shih, Andy et al. (2019). "Smoothing Structured Decomposable Circuits". In: *arXiv preprint arXiv:1906.00311*.