Tractable Probabilistic Models

Representations
Inference
Learning
Applications

Antonio Vergari
University of California, Los Angeles

Nicola Di Mauro
University of Bari

Guy Van den Broeck
University of California, Los Angeles

July 22, 2019 - ***Conference on Uncertainty in Artificial Intelligence (UAI 2019)*** **Tel Aviv**

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

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 $\mathbf{m}$

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

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.?*

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

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

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.?*

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

$$q_1(\mathbf{m}) = p_{\mathbf{m}}(\text{Day} = \text{Mon}, \text{Jam}_{\text{Herzl}} = 1)$$

$\Longrightarrow$ ***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?*

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

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

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?*

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

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

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

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

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

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

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?

  ⇒ *and do we lose something in terms of expressiveness?*

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

  ⇒ *But sometimes approximate inference comes with guarantees (Rina)*

- 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*

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)

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

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

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

pinterest.com/pin/190417890473268205/

Complete evidence queries (EVI)

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

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

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

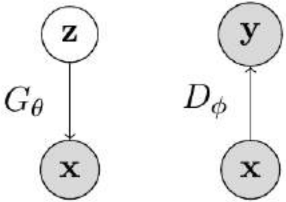
...fundamental in ***maximum likelihood learning***

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

pinterest.com/pin/190417890473268205/

Generative Adversarial Networks

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

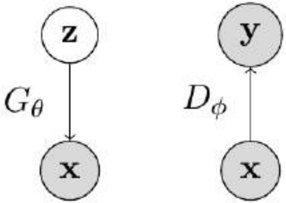


Goodfellow et al., "Generative adversarial nets", 2014

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*

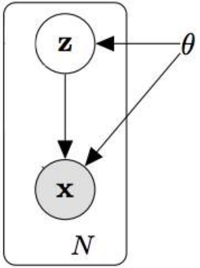


Goodfellow et al., "Generative adversarial nets", 2014

Variational Autoencoders

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

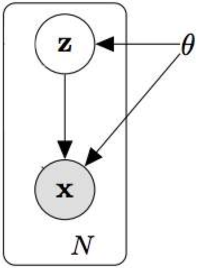
- an explicit likelihood model!



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...

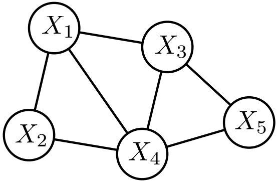
  $\Longrightarrow$ *which is "broken"*

  [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



+

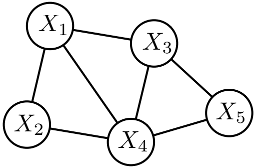
Inference:

- 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)

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



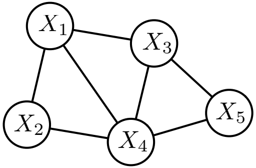
PGMs: MNs and BNs

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!*



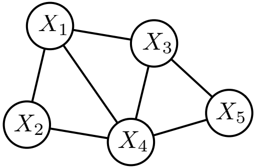
PGMs: MNs and BNs

Markov Networks (MNs)

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

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

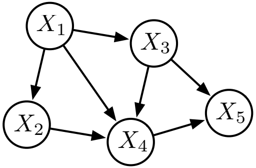
$\Longrightarrow$ EVI *queries are intractable!*



Bayesian Networks (BNs)

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

$\Longrightarrow$ 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.?*

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

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].*

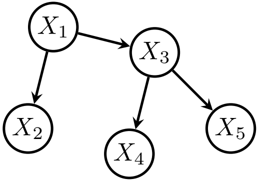
Approximation complexity: Computing MAR and COND approximately within a relative error of $2^{n^{1-\epsilon}}$ for any fixed ϵ is *NP-hard* *[Dagum et al. 1993; Roth 1996].*

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].*

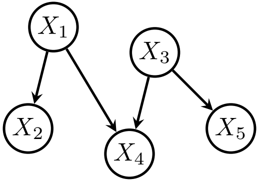
$\Longrightarrow$ *what about bounding the treewidth by design?*

Low-treewidth PGMs



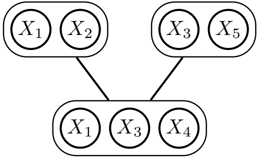
Trees

[Meilă et al. 2000]



Polytrees

[Dasgupta 1999]



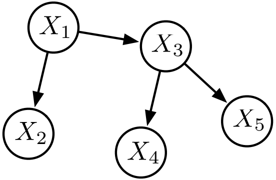
Thin Junction trees

[Bach et al. 2001]

If treewidth is bounded (e.g. ≈ 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: $O(|\mathbf{X}|)$

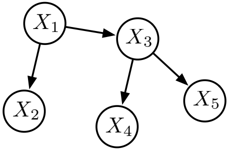
Exact learning from d examples takes $O(|\mathbf{X}|^2 \cdot d)$ with the classical Chow-Liu algorithm[1]

---

[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

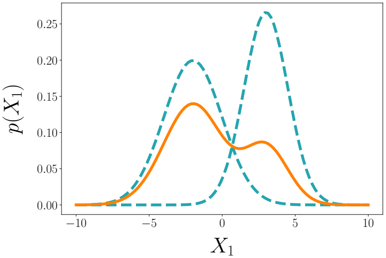


Bounded-treewidth PGMs lose the ability to represent *all possible distributions* ...

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

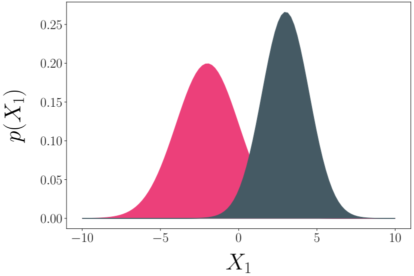


$$p(X) = w_1 \cdot p_1(X) + w_2 \cdot p_2(X)$$

EVI, MAR, CON queries scale linearly in k

Mixtures

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



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

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

$\Longrightarrow$ *increased expressiveness*

Expressiveness and efficiency

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

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

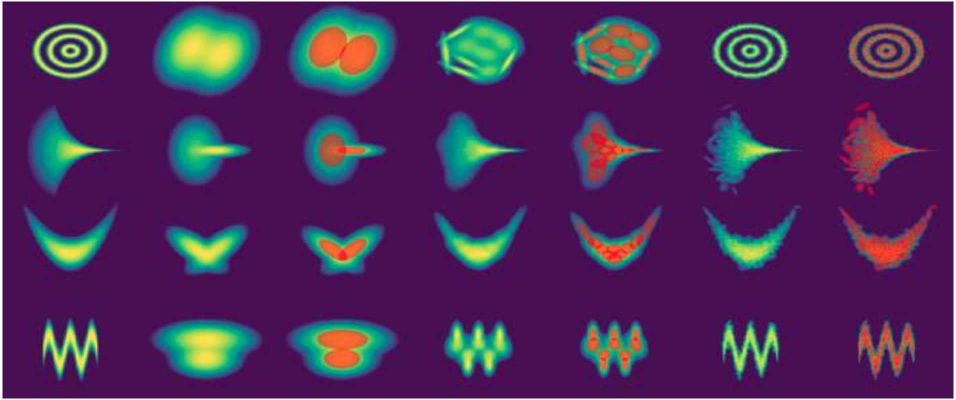
Expressive efficiency (succinctness) compares model sizes in terms of their ability to compactly represent functions

$\Longrightarrow$ *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



$\Rightarrow$ *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?*

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

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?*

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

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

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 BN 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 BN MAP

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

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

pinterest.com/pin/190417890473268205/

Marginal MAP (MMAP)

aka BN MAP

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

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

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

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

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

pinterest.com/pin/190417890473268205/

Marginal MAP (MMAP)

aka BN MAP

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

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

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

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

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

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

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

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

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

pinterest.com/pin/190417890473268205/

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?*

$\Longrightarrow$ **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:

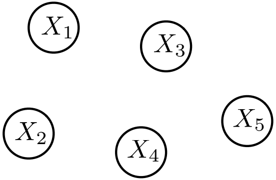
- expected classification agreement *[Oztok et al. 2016; Choi et al. 2017, 2018]*
- expected predictions *[Khosravi et al. 2019a]*

pinterest.com/pin/190417890473268205/

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

Fully factorized models

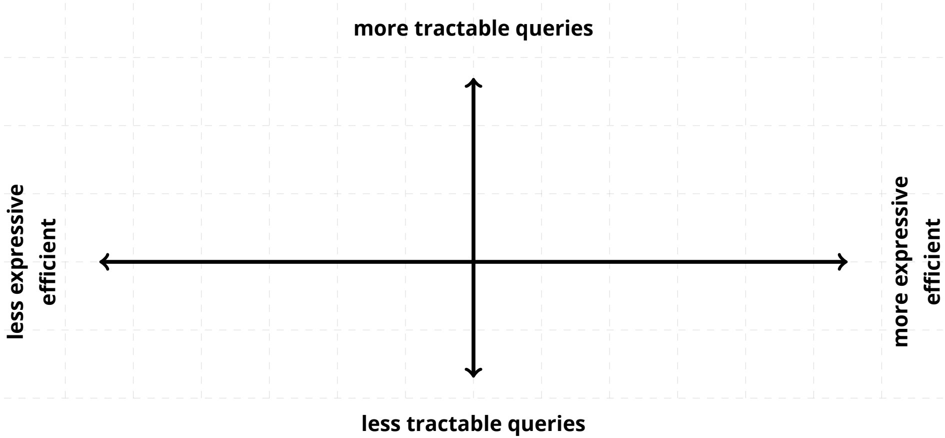
A completely disconnected graph. Example: Product of Bernoullis (PoBs)

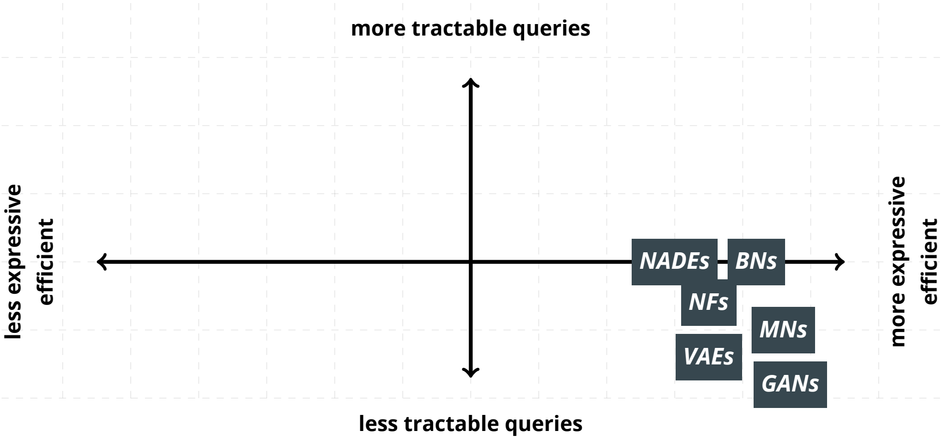


$$p(\mathbf{X}) = \prod_{i=1}^{n} p(x_i|\mathrm{Pa}_{x_i})$$

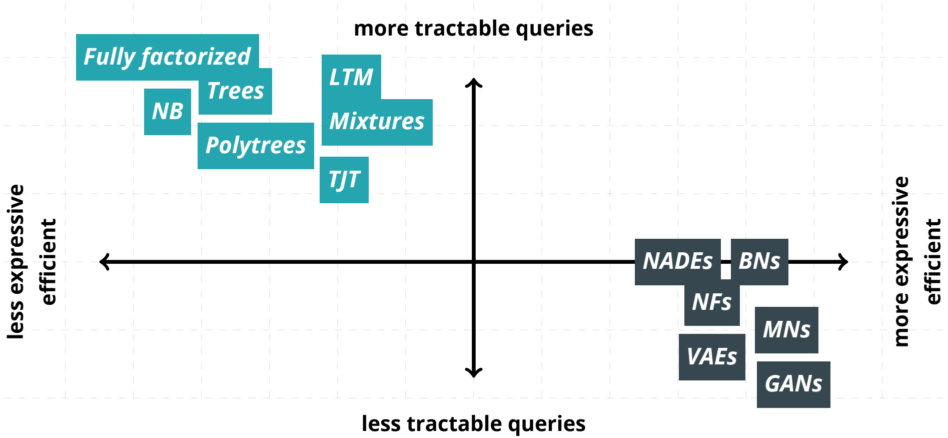
Complete evidence, marginals and MAP, MMAP inference is ***linear***!

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

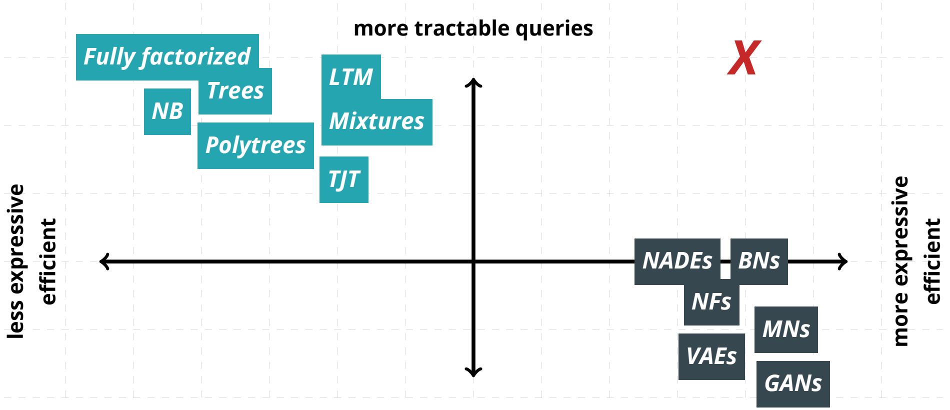

more tractable queries
less expressive
efficient
more expressive
efficient
less tractable queries



Expressive models are not very tractable...



and tractable ones are not very expressive...



probabilistic circuits are at the “sweet spot”

Probabilistic Circuits

Stay Tuned For ...

Next:

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

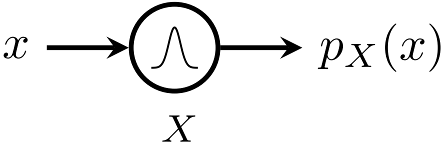
   $\Rightarrow$ *build into a computational graph: a probabilistic circuit*

2. For which queries are probabilistic circuits tractable?

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

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

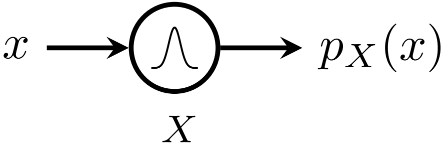
Base Case: Univariate Distributions



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



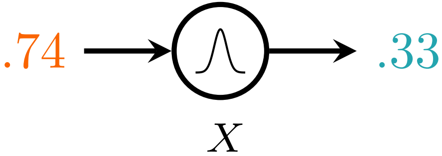
Generally, univariate distributions are tractable for:

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

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

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

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

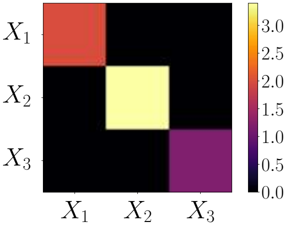
$\Rightarrow$ *often 100% probability for one value of a categorical random variable*

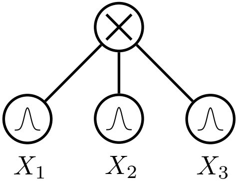
$\Rightarrow$ *for example, X or $\neg X$ for Boolean random variable*

Factorizations are products

Divide and conquer complexity

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



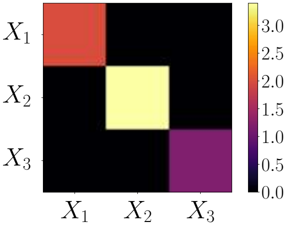


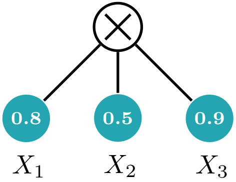
$\Longrightarrow$ *e.g. modeling a multivariate Gaussian with diagonal covariance matrix*

Factorizations are products

Divide and conquer complexity

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



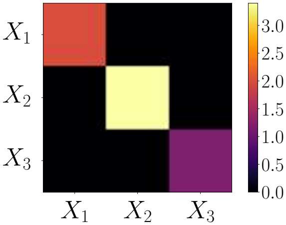


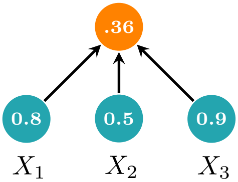
$\Longrightarrow$ *e.g. modeling a multivariate Gaussian with diagonal covariance matrix*

Factorizations are products

Divide and conquer complexity

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

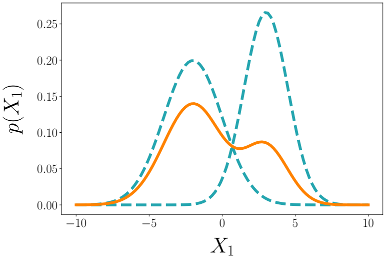




$\Rightarrow$ *e.g. modeling a multivariate Gaussian with diagonal covariance matrix*

Mixtures are sums

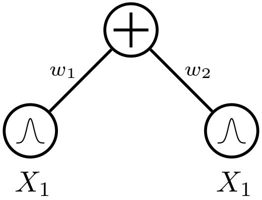
Also mixture models can be treated as a simple ***computational unit*** over distributions



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

Mixtures are sums

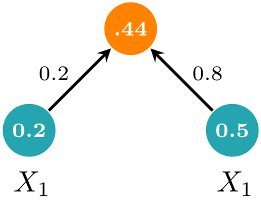
Also mixture models can be treated as a simple ***computational unit*** over distributions



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

Mixtures are sums

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



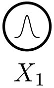
$$p(x) = 0.2 \cdot p_1(x) + 0.8 \cdot p_2(x)$$

With mixtures, we increase expressiveness

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

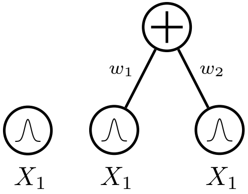
A grammar for tractable models

Recursive semantics of probabilistic circuits



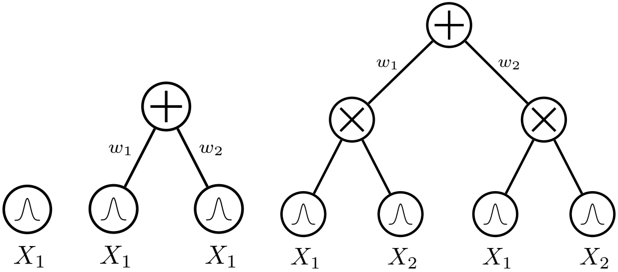
A grammar for tractable models

Recursive semantics of probabilistic circuits



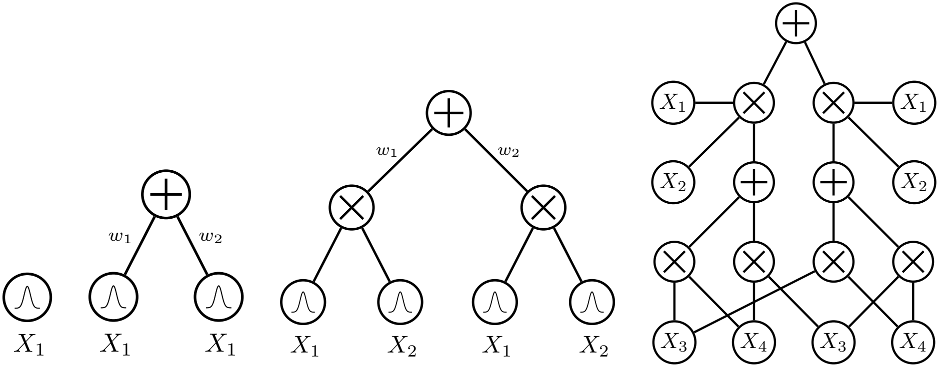
A grammar for tractable models

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*

The perks of being a computational graph

Computations that are repeated can be cached!

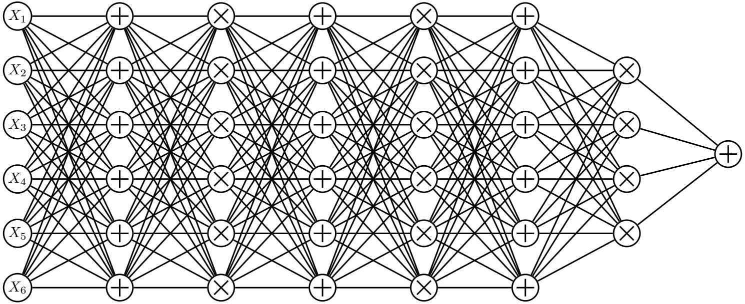
$\Longrightarrow$ *amortizing inference; parameter/structure sharing*

Clear operational semantics! $\Longrightarrow$ *Tractability in terms of circuit size*

Differentiable! $\Longrightarrow$ *gradient-based optimization*

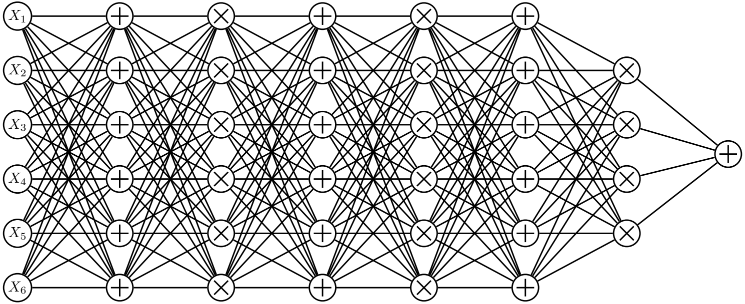
Structural properties on the computational graph cleanly map to tractable query classes...

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!~~

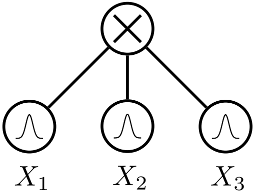
$\Rightarrow$ structural constraints needed for tractability

How do we ensure tractability?

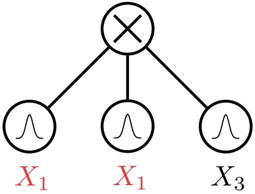
Decomposability

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

⟹ *just like in factorization!*



decomposable circuit



non-decomposable circuit

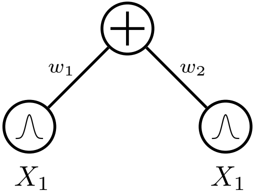
Darwiche et al., "A knowledge compilation map", 2002

Smoothness

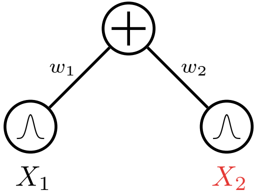
aka completeness

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

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



smooth circuit



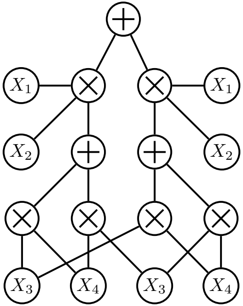
***non*-smooth** circuit

$\Rightarrow$ *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

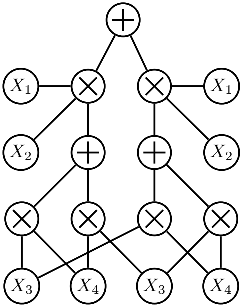
Smoothness and decomposability enable tractable MAR/CON queries

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

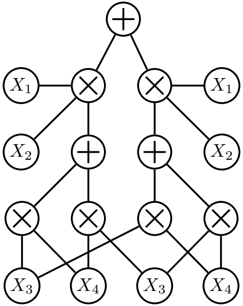
Smoothness and decomposability enable tractable MAR/CON queries

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*



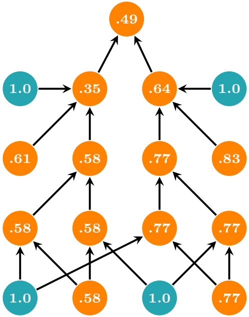
Tractable MAR/CON

Smoothness and decomposability enable tractable MAR/CON queries

Forward pass evaluation $\implies$ *linear in circuit size!*

E.g. to compute $p(X_2, X_3)$, let input distributions over X_1 and X_4 output Z

$\implies$ *for normalized leaf distribution,* 1.0

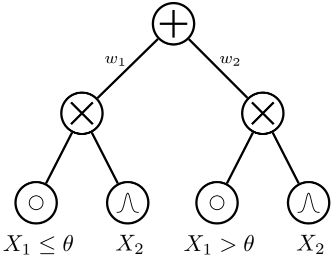


Determinism

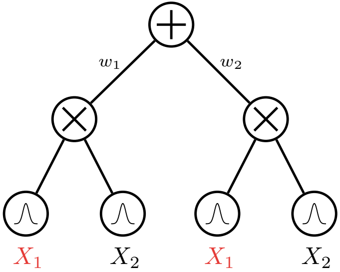
aka selectivity

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*



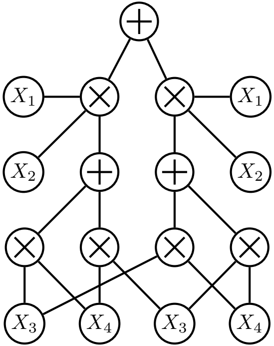
deterministic circuit



non-deterministic circuit

Tractable MAP

The addition of determinism enables tractable MAP queries!



Tractable MAP

The addition of determinism enables tractable MAP queries!

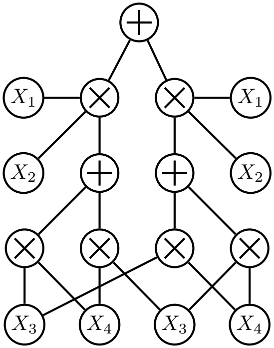
If $p(\mathbf{q}, \mathbf{e}) = p(\mathbf{q_x}, \mathbf{e_x}, \mathbf{q_y}, \mathbf{e_y})$
$= p(\mathbf{q_x}, \mathbf{e_x}) p(\mathbf{q_y}, \mathbf{e_y})$ (***decomposable*** product node):

$$\underset{\mathbf{q}}{\operatorname{argmax}}\, p(\mathbf{q} \mid \mathbf{e}) = \underset{\mathbf{q}}{\operatorname{argmax}}\, p(\mathbf{q}, \mathbf{e}) =$$

$$\underset{\mathbf{q_x}, \mathbf{q_y}}{\operatorname{argmax}}\, p(\mathbf{q_x}, \mathbf{e_x}, \mathbf{q_y}, \mathbf{e_y}) =$$

$$\underset{\mathbf{q_x}}{\operatorname{argmax}}\, p(\mathbf{q_x}, \mathbf{e_x}), \underset{\mathbf{q_y}}{\operatorname{argmax}}\, p(\mathbf{q_y}, \mathbf{e_y})$$

$\Longrightarrow$ *solving optimization independently*



Tractable MAP

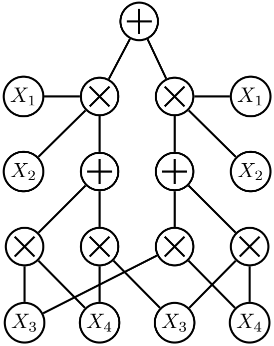
The addition of determinism enables tractable MAP queries!
If $p(\mathbf{q}, \mathbf{e}) = \sum_i w_i p_i(\mathbf{q}, \mathbf{e}) = w_c p_c(\mathbf{q}, \mathbf{e})$,
(***deterministic*** sum node):

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

$$\operatorname*{argmax}_{\mathbf{q}} \max_i w_i p_i(\mathbf{q}, \mathbf{e}) =$$

$$\operatorname*{argmax}_{\mathbf{q}} w_c p_c(\mathbf{q}, \mathbf{e})$$

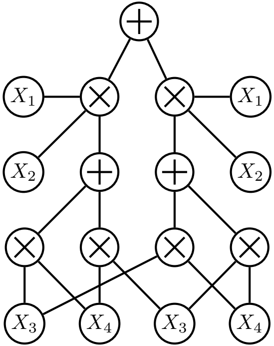
$\Longrightarrow$ *only one non-zero children* c



Tractable MAP

The addition of determinism enables tractable MAP queries!

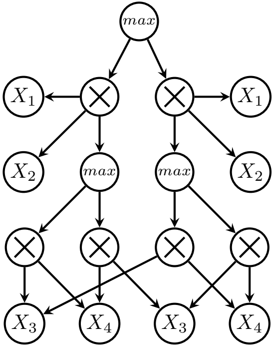
Evaluating the circuit twice:
bottom-up and ***top-down*** $\Longrightarrow$ *still linear in circuit size!*



Tractable MAP

The addition of determinism enables tractable MAP queries!

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

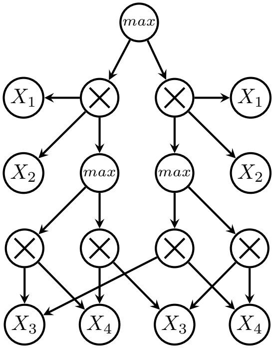
Tractable MAP

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



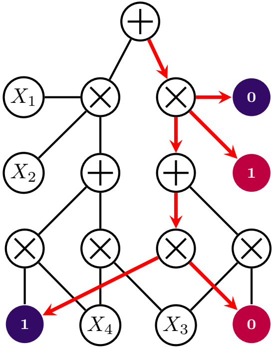
Tractable MAP

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



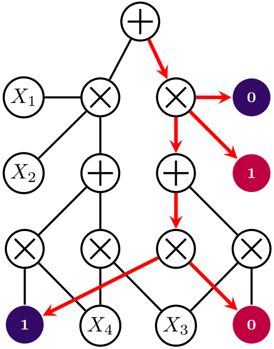
Tractable MAP

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* $\mathbf{Z}$

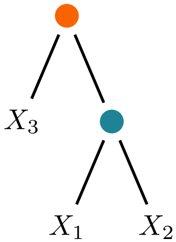
$$\operatorname*{argmax}_{\mathbf{q}} \sum_i w_i p_i(\mathbf{q}, \mathbf{e}) = \operatorname*{argmax}_{\mathbf{q}} \sum_{\mathbf{z}} p(\mathbf{q}, \mathbf{z}, \mathbf{e}) \neq \operatorname*{argmax}_{\mathbf{q}} \max_{\mathbf{z}} p(\mathbf{q}, \mathbf{z}, \mathbf{e})$$

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

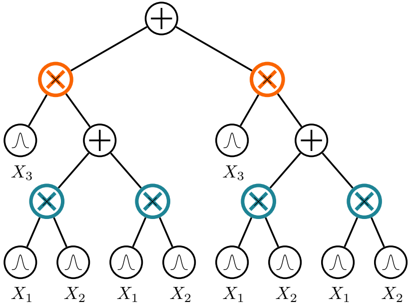
Structured decomposability

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

$\Rightarrow$ *stronger requirement than decomposability*



vtree

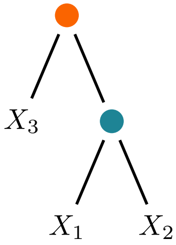


structured decomposable circuit

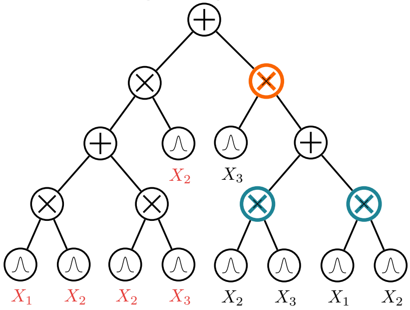
Structured decomposability

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

$\Longrightarrow$ *stronger requirement than decomposability*



vtree



non 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:

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

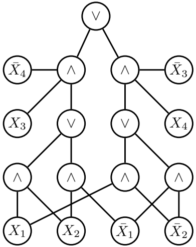
$\Longrightarrow$ *Algebraic model counting [Kimmig et al. 2017], Semi-ring programming [Belle et al. 2016]*

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

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

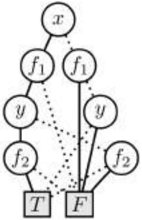
$\Longrightarrow$ *logical circuits!*

Logical circuits



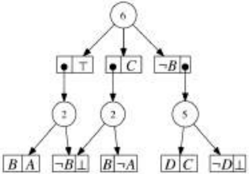
s/d-D/DNFs

[Darwiche et al. 2002]



O/BDDs

[Bryant 1986]



SDDs

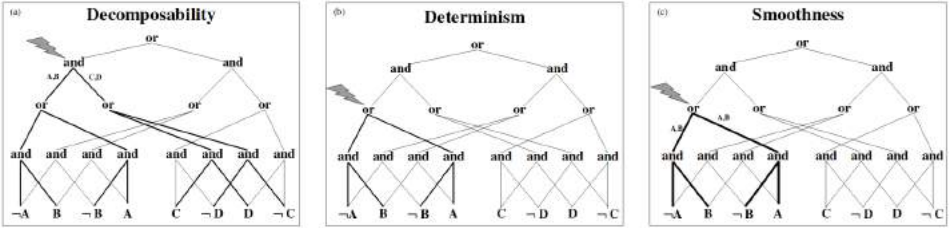
[Darwiche 2011]

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

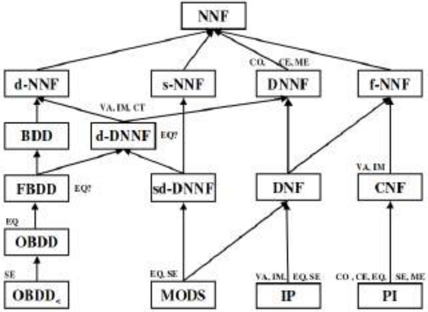


Darwiche et al., "A knowledge compilation map", 2002

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**)

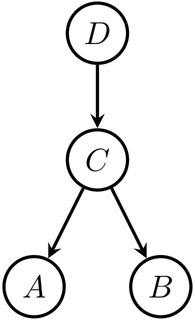
$$\mathrm{WMC}(\Delta, w) = \sum_{\mathbf{x} \models \Delta} \prod_{l \in \mathbf{x}} w(l)$$

- 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

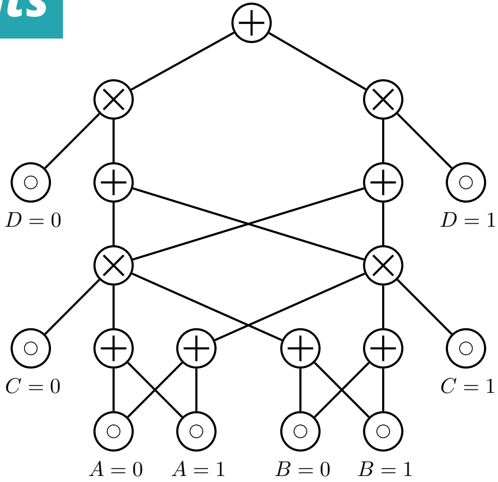
- End result equivalent to probabilistic circuit: efficiently replace parameter variables in logical circuit by edge parameters in probabilistic circuit

From trees to circuits

via compilation

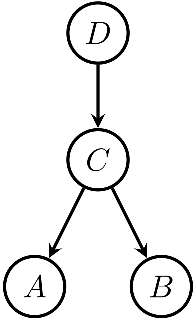


$\rightarrow$



From trees to circuits

via compilation

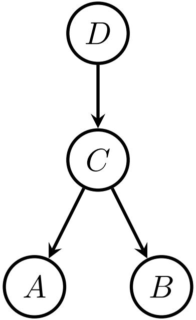


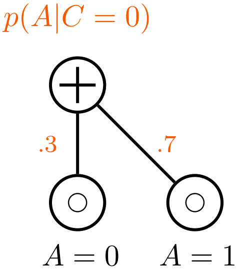
Bottom-up ***compilation***: starting from leaves...

From trees to circuits

via compilation

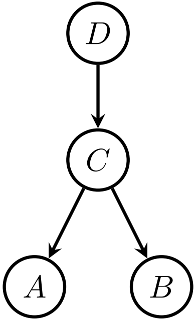
...compile a leaf CPT





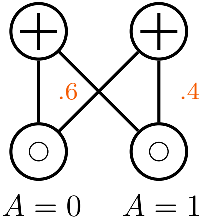
From trees to circuits

via compilation



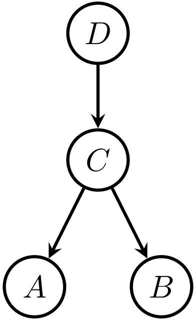
...compile a leaf CPT

$p(A|C = 1)$

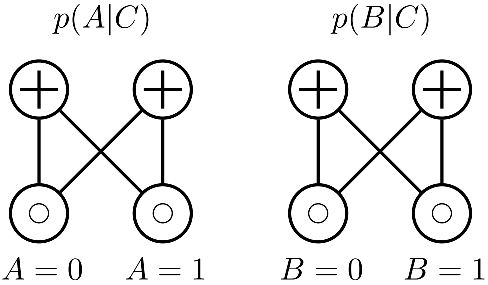


From trees to circuits

via compilation

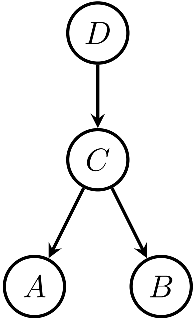


...compile a leaf CPT...for all leaves...

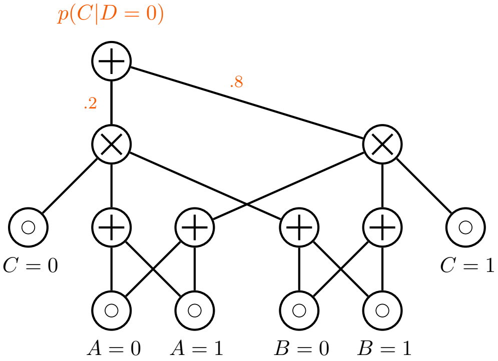


From trees to circuits

via compilation

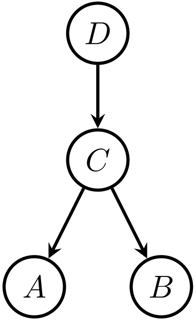


...and recurse over parents...

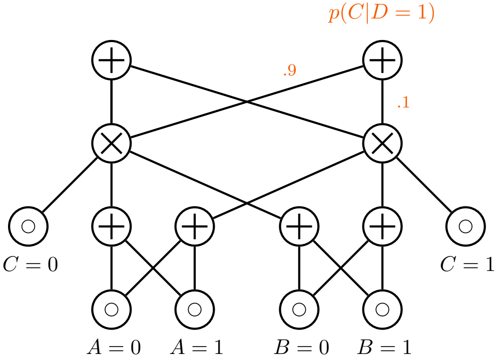


From trees to circuits

via compilation

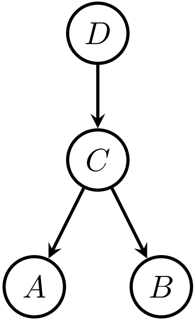


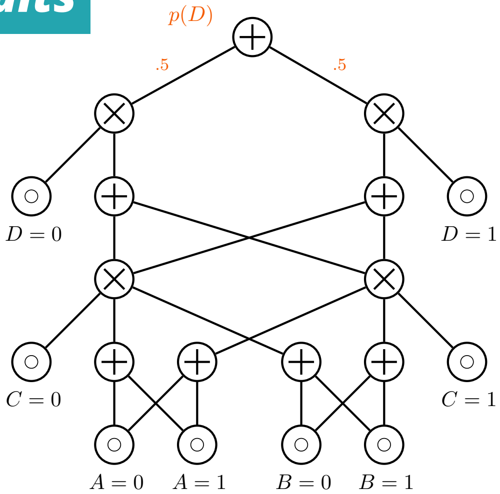
...while reusing previously compiled nodes!...



From trees to circuits

via compilation





Low-treewidh PGMs

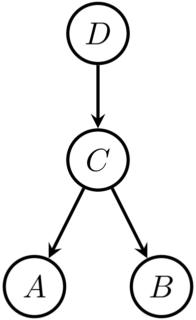
Tree, polytrees and
Thin Junction trees
can be turned into

- decomposable
- smooth
- deterministic

circuits

Therefore they support
tractable

- EVI
- MAR/CON
- MAP



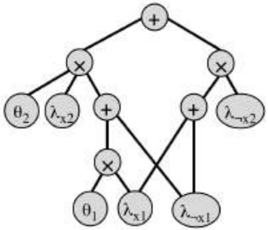
Arithmetic Circuits (ACs)

ACs *[Darwiche 2003]* are

- decomposable
- smooth
- deterministic

They support tractable

- EVI
- MAR/CON
- MAP



⇒ *parameters are attached to the leaves*

⇒ *...but can be moved to the sum node edges [Rooshenas et al. 2014]*

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

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

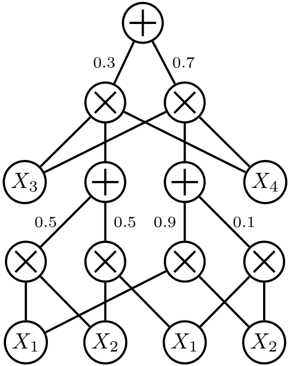
Sum-Product Networks (SPNs)

SPNs *[Poon et al. 2011]* are

- decomposable
- smooth
- ~~deterministic~~

They support tractable

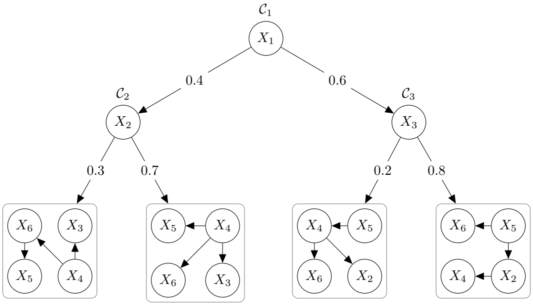
- EVI
- MAR/CON
- ~~MAP~~



⇒ *deterministic SPNs are also called* selective *[Peharz et al. 2014a]*

Cutset Networks (CNets)

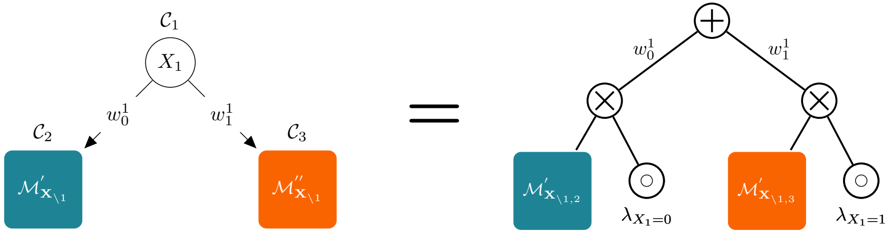
A CNet *[Rahman et al. 2014]* is a ***weighted model-trees*** *[Dechter et al. 2007]* whose leaves are tree Bayesian networks



⟹ *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!*

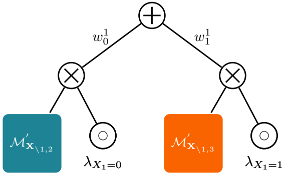
CNets as probabilistic circuits

CNets are

- decomposable
- smooth
- deterministic

They support tractable

- EVI
- MAR/CON
- MAP



$\Longrightarrow$ EVI *can be computed in* $O(|\mathbf{X}|)$

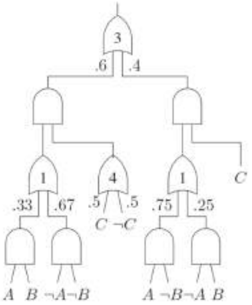
Probabilistic Sentential Decision Diagrams

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

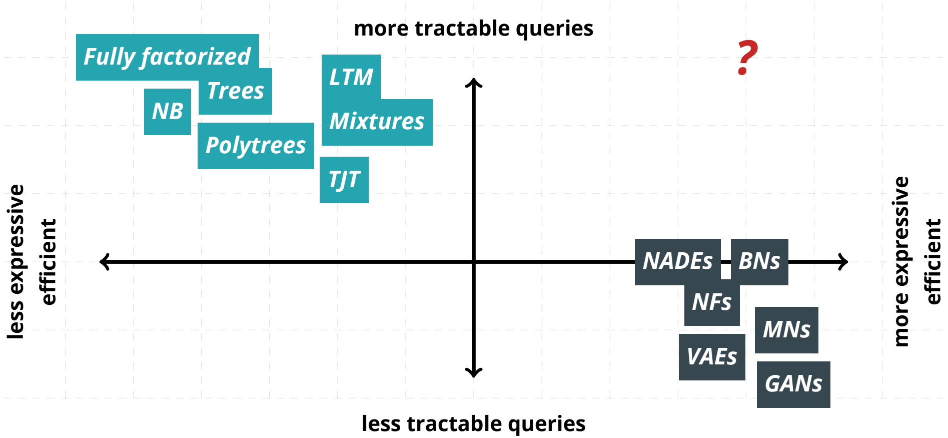
- structured decomposable
- smooth
- deterministic

They support tractable

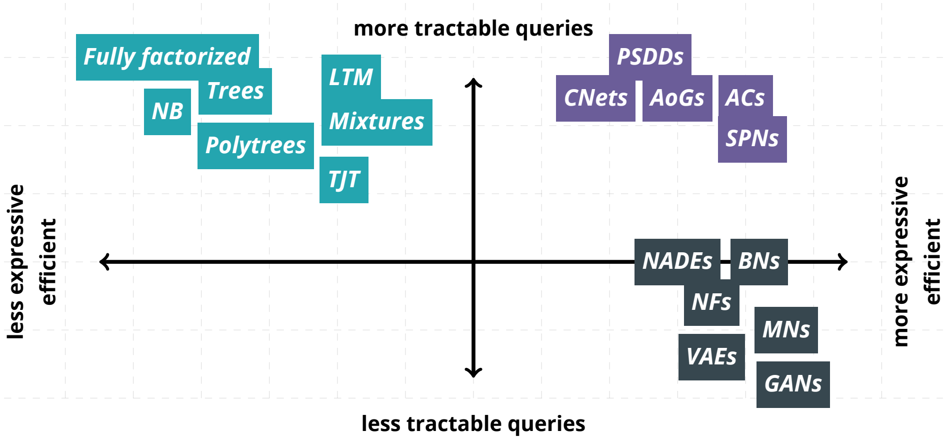
- EVI
- MAR/CON
- MAP
- Complex queries!



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

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, ...*

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

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

After: *What is this used for?*

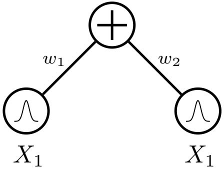
Learning circuit parameters

Sum node distibution $p(\mathbf{X})$ can be interpreted as a marginal distribution of $p(\mathbf{X}, Z)$ over $\mathbf{X}$ and a latent variable Z

- $p(\mathbf{X}|Z = k)$ **child distribution**
- $p(Z = k) = w_k$ **weight**

Even leaf distributions could be parametrized by $\boldsymbol{\theta}$

Learning parameters involves learning both sum and leaf parameters $(\boldsymbol{w}, \boldsymbol{\theta})$



Learning circuit parameters

deterministic
circuits

⇒ closed-form, convex optimization
[Kisa et al. 2014b; Liang et al. 2019]

non- deterministic
circuits

⇒

- 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 and a complete dataset $\mathbf{D}$,
maximize the likelihood of parameters given examples in the dataset

$$\theta^{\mathsf{MLE}} = \underset{\theta}{\operatorname{argmax}}\, L(\theta; \mathbf{D}) = \underset{\theta}{\operatorname{argmax}} \prod_i p_\theta(\mathbf{d}_i)$$

With determinism, L decomposes over the parameters, and θ^{MLE} has a closed-form solution

$\Longrightarrow$ *compute sufficient statistics (just count)*

$\Longrightarrow$ *a single pass of the dataset required!*

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

Hard/Soft Parameter Updating

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

Hard/Soft Parameter Updating

Expectation Maximization

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

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

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*

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

- the posterior is intractable
  - assuming a prior $p(\mathbf{w}) = \prod_{i \in sumNodes} Dir(\mathbf{w}_i|\alpha_i)$
    - considering circuits with normalized weights $w_{ij} \geq 0$ and $\sum_j w_{ij} = 1, \forall i \in sumNodes$
  - the posterior becomes a mixture of products of Dirichlets
  - the number of mixture components is exponential in the number of sum nodes

Bayesian Parameter Learning

Moment matching (oBMM) : approximate the posterior after each update with a tractable distribution that matches some moments of the exact, but intractable posterior

- the joint $p(\mathbf{w})$ is approximated by a product of Dirichlets
- the first and second moment of each marginal $p(\mathbf{w}_i)$ are used to set the hyperparameters α_i of each Dirichlet in the product of Dirichlets

oBMM extended to continuous models with Gaussian leaves

CVB-SPN: a collapsed variational inference algorithm

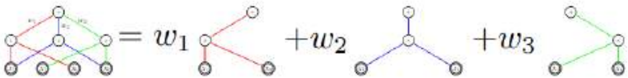
- better results than oBMM

Rashwan et al., "Online and Distributed Bayesian Moment Matching for Parameter Learning in Sum-Product Networks", 2016
Jaini et al., "Online Algorithms for Sum-Product Networks with Continuous Variables", 2016
Zhao et al., "Collapsed Variational Inference for Sum-Product Networks", 2016

Parameter Learning

Sequential monomial approximation & Concave-convex procedure

Any complete and decomposable circuit is equivalent to a mixture of trees where each tree corresponds to a product of univariate distributions



- learning the parameters based on the MLE principle can be formulated as a **signomial program** *Sequential Monomial Approximation (SMA)*
- the signomial program formulation can be equivalently transformed into a difference of convex functions *Concave-convex Procedure (CCCP)*

Zhao et al., "A Unified Approach for Learning the Parameters of Sum-Product Networks", 2016

Structure learning

Greedy layerwise

LearnSPN& and variants

Structure learning as search

defining operators

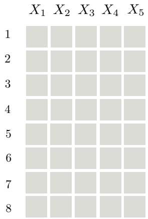
Local search

LearnPSDD

Random structures

XCNets, RAT-SPNs

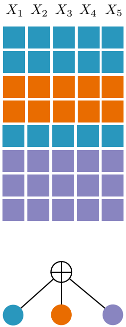
LearnSPN



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

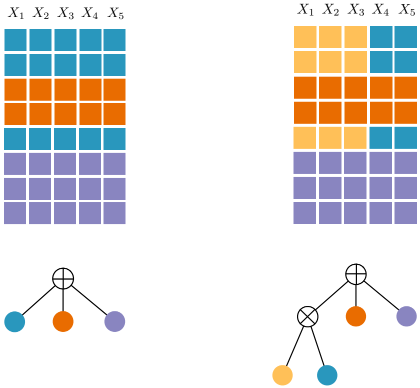
LearnSPN



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

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

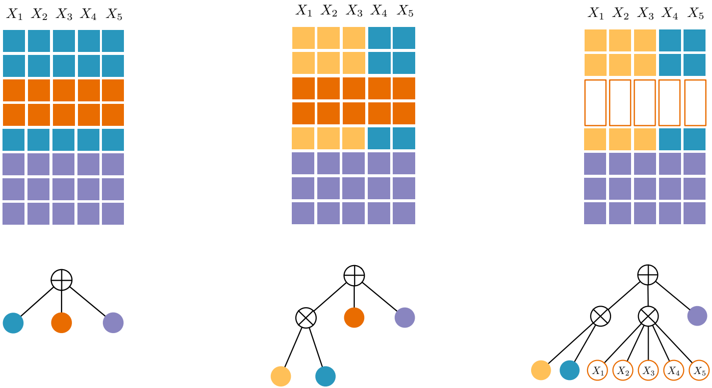
LearnSPN



...***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

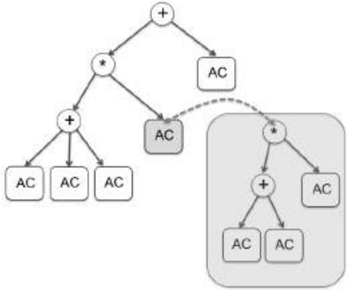
LearnSPN variants

- **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. 2018a]* 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]*

ID-SPN

ID-SPN works like LearnSPN: clustering instance and variables for sum and product nodes

- start with a single AC representing a *tractable Markov network*
- stop the process before reaching univariate distributions
  - learn a *tractable MN represented by an AC* factorizing a multivariate distribution



$\Longrightarrow$ *SPNs with tractable multivariate distributions as leaves—MN ACs*

Rooshenas et al., "Learning Sum-Product Networks with Direct and Indirect Variable Interactions", 2014

Other variants

Bottom up learning *[Peharz et al. 2013]*

- starting from simple models over small variable scopes
- growing models over larger variable scopes, building successively more expressive models guided by dependence tests and a maximum mutual information principle

Greedy for deterministic circuits *[Peharz et al. 2014a]*

- hill climbing tranforming a network with split and merge operations

Graph SPNs from tree SPNs by merging similar sub-structures *[Rahman et al. 2016b]*

- bottom-up merging sub-SPNs with similar distributions defined over the same variables

Cut(e)set Network

For deterministic circuits, structure scores decompose

CNet likelihood decomposition

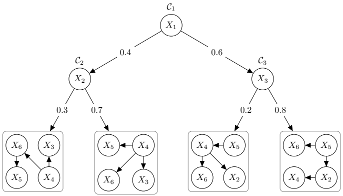
- $\mathcal{L}_{\mathcal{D}}(G;\boldsymbol{\theta}) = \sum_i \alpha_i + \mathcal{L}_{\mathcal{D}_i}(G_i;\boldsymbol{\theta_i})$

BIC score decomposition

- $\mathcal{L}(G';\boldsymbol{\theta}') - \mathcal{L}(G';\boldsymbol{\theta}') > (\log M)/2$

Structure Learning

- start with a single tractable multivariate model (CLT)
- substitute a leaf node with the best CNet improving both the LL and the BIC score



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

PSDD Structure Learning

Learning vtree

A variable tree (vtree)

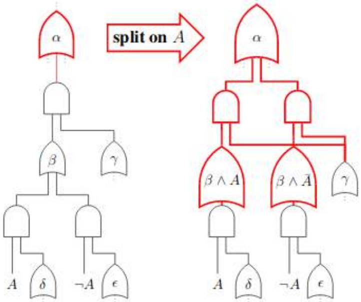
- a full binary tree
- leaves are labeled with variables
- internal vtree nodes split variables into those appearing in the left subtree $\mathbf{X}$ and those in the right subtree $\mathbf{Y}$
- it can be learned from data in a top-down or bottom-up fashion

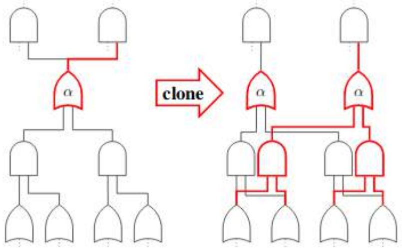
$\Longrightarrow$ *maximising pairwise MI instead of joint MI*

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

PSDD Structure Learning

Local operations





- incrementally change the PSDD structure preserving syntactic soundness

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

LearnPSDD

LearnPSDD incrementally improves the structure of a PSDD to better fit the data

- in every step, the structure is changed by executing an operation
- learning continues until the log-likelihood on validation data stagnates, or a desired time or size limit is reached
- the operation to execute is greedily chosen based on the best likelihood improvement per size increment

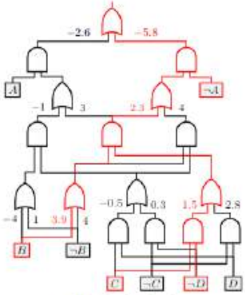
$$\text{score} = \frac{\log \mathcal{L}(r'|\mathcal{D}) - \log \mathcal{L}(r|\mathcal{D})}{\text{size}(r') - \text{size}(r')}$$

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

Learning Logistic Circuits

- propagates values and parameters bottom-up
- logistic function at root node with weight function $g_r(\mathbf{x})$

$$Pr(Y=1|\mathbf{x}) = \frac{1}{1+\exp(-g_r(\mathbf{x}))}$$



(a) Logistic circuit

A	B	C	D	$g_r(ABCD)$	$\Pr(Y=1 \mid ABCD)$
1	0	1	1	-3.1	4.31%
0	1	1	0	1.9	86.99%
1	1	1	0	5.8	99.70%

(b) Weights and classification probabilities for select examples

Liang et al., "Learning Logistic Circuits", 2019

Learning Logistic Circuits

Parameter Learning

- Due to decomposability and determinism, any logistic circuit model can be reduced to a logistic regression model over a particular feature set

$$Pr(Y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\overline{\mathbf{X}}\theta)}$$

  - $\overline{\mathbf{X}}$ is some vector of features extracted from the raw example $\mathbf{X}$

Structure Learning

- use the split operation like in LearnSPDD

Liang et al., "Learning Logistic Circuits", 2019

Bayesian Structure Learning

A prior distribution for SPN trees

The priors are defined recursively, node by node

- prior of each sum-node s is a Dirichlet process, with concentration parameter α_s and base distribution $G_P(s)$
  - $G_P(s)$: probability distribution over the set of possible product nodes with scope s
- the prior distribution over SPNs is specified as a tree of Dirichlet Processes over product nodes

The model is straightforward altered for DAG using hierarchical Dirichlet Process

Lee et al., “Non-Parametric Bayesian Sum-Product Networks”, 2014

ABDA

Automatic Bayesian Density Analysis

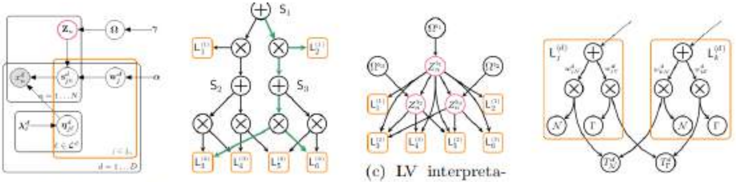
Overcoming the problem in DE of assuming *homogeneous* RVs and *shallow* dependency structures

- ABDA relies on SPNs to capture statistical dependencies in heterogeneous data at different granularity through a hierarchical co-clustering
  - inference for both the statistical data types and (parametric) likelihood models
  - robust estimation of missing values
  - detection of corrupt or anomalous data
  - automatic discovery of the statistical dependencies and local correlations in the data

Vergari et al., "Automatic Bayesian Density Analysis", 2018

ABDA

Generative model



(a) Graphical model (b) SPN (c) LV interpretation (d) Type-augmented SPN

$$Z_n^{\mathsf{S}} \sim \mathsf{Cat}(\Omega^{\mathsf{S}}), \Omega^{\mathsf{S}} \sim \mathsf{Dir}(\boldsymbol{\gamma})$$

$$\mathbf{w}_j^d \sim \mathsf{Dir}(\boldsymbol{\alpha}), s_{j,n}^d \sim \mathsf{Cat}(\mathbf{w}_j^d)$$

prior on $\boldsymbol{\eta}_{j,l}^d$ parametrized with $\boldsymbol{\lambda}_l^d$

Vergari et al., "Automatic Bayesian Density Analysis", 2018

Bayesian SPNs

Learning both the structure and parameters

A well-principled Bayesian approach to SPN learning, simultaneously over both structure and parameters

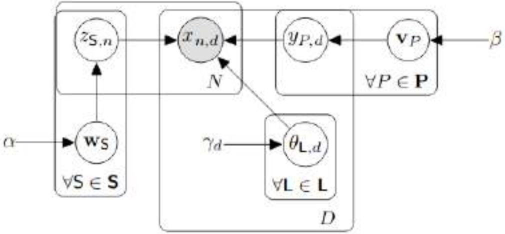
- the structure learning problem is decomposes into two steps
  1. proposing a computational graph
     $\Rightarrow$ *laying out the arrangement of sums, products and leaf distributions*
  2. learning the scope-function, which assigns to each node its scope

---

Trapp et al., "Bayesian Learning of Sum-Product Networks", 2019

Bayesian SPNs

Generative model



$$\mathbf{w}_S \mid \alpha \sim Dir(\mathbf{w}_S \mid \alpha) \;\; \forall S, \qquad z_{S,n} \mid \mathbf{w}_S \sim Cat(z_{S,n} \mid \mathbf{w}_S) \;\; \forall S \, \forall n,$$

$$\theta_L \mid \gamma \sim p(\theta_L \mid \gamma) \;\; \forall L, \qquad \mathbf{x}_n \mid \mathbf{z}_n, \theta \sim \prod_{L \in T(\mathbf{z}_n)} L(\mathbf{x}_{L,n} \mid \theta_L) \;\; \forall n.$$

Trapp et al., "Bayesian Learning of Sum-Product Networks", 2019

Randomized structure learning: RAT-SPNs

Random Tensorized SPNs (RAT-SPNs)

- SPNs are obtained by first constructing a random region graph
- subsequently populating the region graph with tensors of SPN nodes
- implemented in Tensorflow and easily optimized using automatic differentiation, SGD, and automatic GPU-parallelization
- implementing an SPN dropout heuristic
    - an elegant probabilistic interpretation as marginalization of missing features (dropout at inputs) and as injection of discrete noise (dropout at sum nodes)
- comparable DNNs; complete joint distribution over variables; robust in the presence of missing features; well-calibrated uncertainty estimates over their inputs

---

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

RAT-SPNs

Losses

Generative training (EM): $\text{LL} = \frac{1}{N}\sum_{i=1}^{N} \log \mathcal{S}(\mathbf{x}_n)$

Discriminative training (SGD): $\text{CE} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\mathcal{S}_{y_n}(\mathbf{x}_n)}{\sum_{y'} S_{y'}(\mathbf{x}_n)}$

Hybrid training (SGD): $\text{O} = \lambda\text{CE} - (1-\lambda)\frac{\text{LL}}{|\mathbf{X}|}$

More details and results during the UAI oral session, tomorrow at 2:30pm

Ensembles of Probabilistic Circuits

To mitigate issues like the scarce accuracy of a single model and their tendency to overfit, circuits can be employed as the components of a mixture

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

Employing EM to alternatively learn both the weights and the mixture components

- issues about convergence and instability of EM $\rightarrow$ **impractical**

Ensembles of Probabilistic Circuits

Bagging Probabilistic Circuits

- 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 that bagging

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

Ensembles of Probabilistic Circuits

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

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

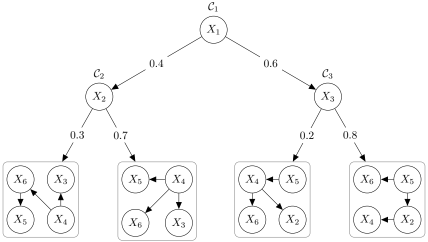
- GBDE: a kernel based generalization of BDE—AdaBoost style algorithm
- sequential EM
  - at each step m, jointly optimize η_m and c_m keeping f_{m-1} fixed

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

Extremely Randomized CNets: XCNets

Learning both the structure and parameters of a CNet equals to perform searching in the space of all probabilistic weighted model trees

- a problem tackled in a *two-stage greedy fashion*
  1. performing a top-down search in the space of weighted OR trees
  2. learning TPMs as leaf distributions



XCNets

LearnCNet($\mathcal{D}$, $\mathbf{X}$, α, δ, σ)

```
Input: a dataset 𝒟 over RVs X; α: δ min number samples; σ min number features
Output: a CNet 𝒞 encoding p_𝒞(X) learned from 𝒟
if |𝒟| > δ and |X| > σ then
    X_i ← select(𝒟, X, α)                                  ▷ select the RV to condition on
    𝒟_0 ← {ξ ∈ 𝒟 : ξ[X_i] = 0}, 𝒟_1 ← {ξ ∈ 𝒟 : ξ[X_i] = 1}
    w_0 ← |𝒟_0|/|𝒟|, w_1 ← |𝒟_1|/|𝒟|
    𝒞 ← w_0 · LearnCNet(𝒟_0, X_\i, α, δ, σ) + w_1 · LearnCNet(𝒟_1, X_\i, α, δ, σ)
else
    𝒞 ← learnLeafDistribution(𝒟, X, α)
return 𝒞
```

XCNets (Extremely Randomized CNets): select chooses one RV at random

Di Mauro et al., "Fast and Accurate Density Estimation with Extremely Randomized Cutset Networks", 2017

Online Learning

Discrete data *[Lee et al. 2013]*

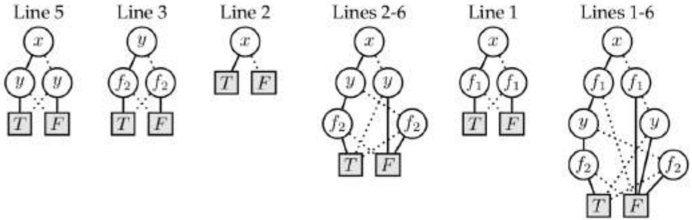
- a variant of LearnSPN using online clustering
- sum nodes can be extended with more children
- product nodes are never modified

Continuos data *[Hsu et al. 2017]*

- starting with a network assuming all variables independent
- correlation are incrementally introduced in the form of a multivariate Gaussian or a mixture distribution

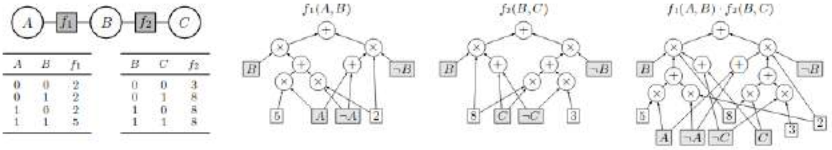
Knowledge Compilation

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



Knowledge Compilation

Compilation to arithmetic circuits



A	B	f_1
0	0	2
0	1	2
1	0	2
1	1	5

B	C	f_2
0	0	3
0	1	8
1	0	8
1	1	8

- the joint distribution $P(A, B, C)$ can be represented as an AC
- the AC has inputs variable assignements (A and $\neg A$) or constants
- internal nodes are sums or product
- *complete assignment*: set variable assignments to 1 (opposing to 0)
  - the root of the AC evaluates the weight (unnormalized probability) of that world

---

Darwiche, Modeling and Reasoning with Bayesian Networks, *2009*

Hybridizing TPMs with intractable models

Collapsed compilation

Inference algorithms based on a knowledge compilation approach perform exact inference by compiling a worst-case exponentially-sized arithmetic circuit representation

- *online collapsed importance sampling*
  - choosing which variable to sample next based on the values sampled for previous variables
- *collapsed compilation*
  - maintaining a partially compiled arithmetic circuit during online collapsed sampling

---

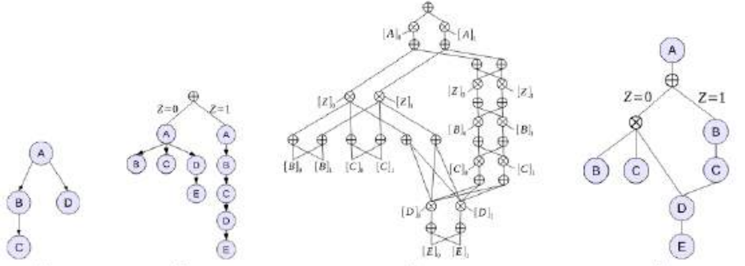
Friedman et al., "Approximate Knowledge Compilation by Online Collapsed Importance Sampling", 2018

Hybridizing TPMs with intractable models

Sum-Product Graphical Model (SPGM)

A probabilistic architecture combining SPNs and Graphical Models (GMs)

$\Rightarrow$ *tractable inference (SPN) + high-level abstraction (PGM)*



(a) (b) (c) (d)

Desana et al., "Sum–product graphical models", 2019

Hybridizing TPMs with intractable models

sum-product VAE

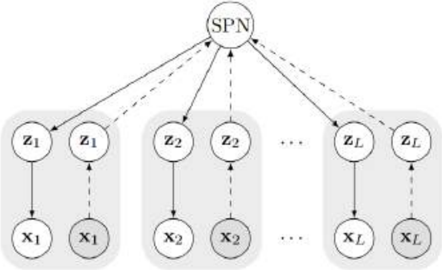


Table 1. Performance on test set, 5000-sample IWAE ELBO

	Continuous			Discrete		
	mnist	svhn	cifar	mnist	svhn	cifar
SPVAE	**2819**	**1936**	**1283**	**-1532**	**-3891**	**-5543**
VAE	2598	1442	896	-2351	-4965	-7200
Conv-SPVAE	2702	**2101**	**1397**	**-927**	**-3666**	**-4562**
Conv-VAE	**2907**	1896	1191	-2099	-4115	-6752

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

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

20 Datasets

current state-of-the-art

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]*

Challenge #1

better benchmarks

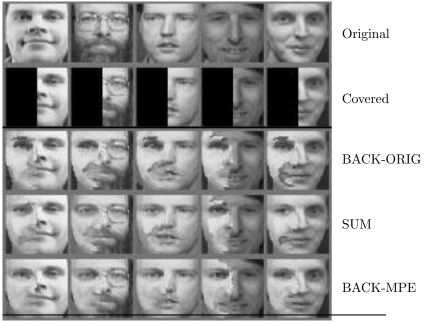
Move beyond toy benchmarks

to datasets reflecting

the **complex** *and* **heterogeneous** *nature of* **real data**!

Computer vision

Image reconstruction and ***inpainting*** $\Longrightarrow$ MAP inference

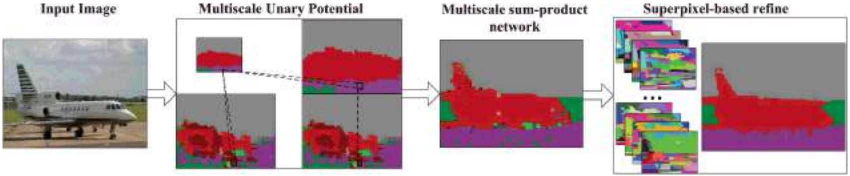


Reconstructing some symmetries (eyes, but not beards, glasses).

Good results for 2001...

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

Image segmentation

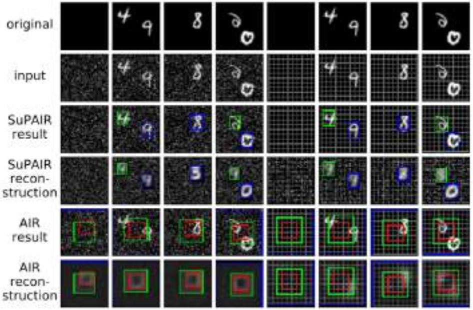


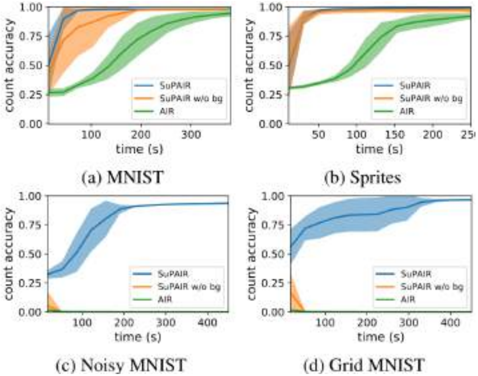
Semantic segmentation is again MAP inference!

Even approximate MAP for non-deterministic circuits (SPNs) has 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

Scene Understanding: Su-PAIR





Stelzner et al., *“Faster Attend-Infer-Repeat with Tractable Probabilistic Models”, 2019*

Challenge #2

hybridizing tractable and intractable models

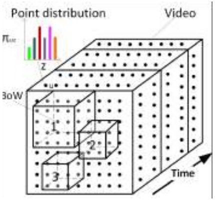
Hybridize probabilistic inference*:*

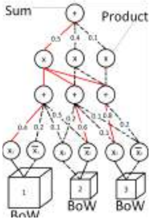
tractable models inside intractable loops

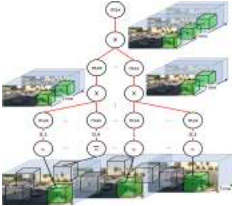
and intractable small boxes glued by tractable inference!

Activity recognition

Exploiting part-based decomposability along pixels *and time* (frames). Probabilistic circuits for MAP and MMAP inference and explanations.



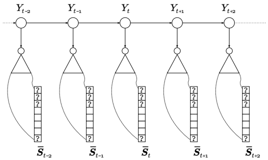


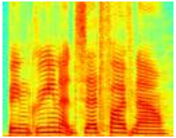


Amer et al., "Sum Product Networks for Activity Recognition", 2015
Wang et al., "Hierarchical spatial sum–product networks for action recognition in still images", 2016
Chiradeep Roy et al., "Explainable Activity Recognition in Videos using Dynamic Cutset Networks", 2019

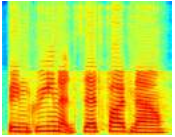
Speech reconstruction and extension

Probabilistic circuits to model the joint pdf of ***observables in HMMs*** (HMM-SPNs), again leveraging tractable inference: marginals and MAP

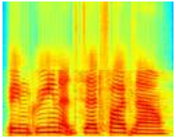




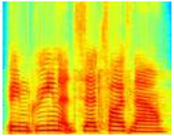
(a) Original full bandwidth



(b) Reconstruction HMM-LP



(c) Reconstruction HMM-GMM

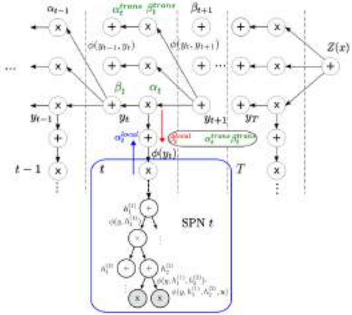


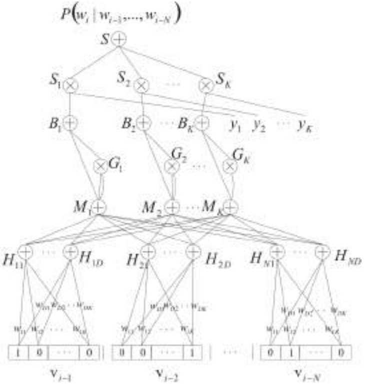
(d) Reconstruction HMM-SPN

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

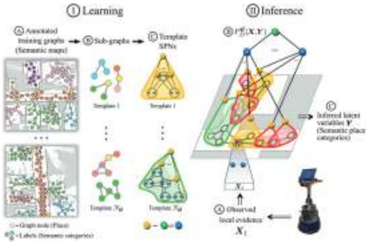
Sequence labeling





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

Robotics

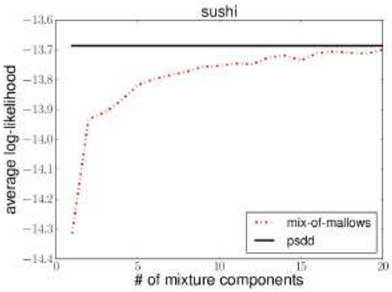


Hierarchical planning robot executions

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

SOP: Preference learning



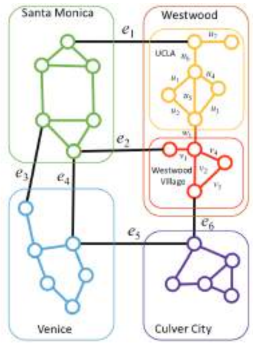
Preferences and rankings as logical constraints

Structured decomposable circuits for advanced queries

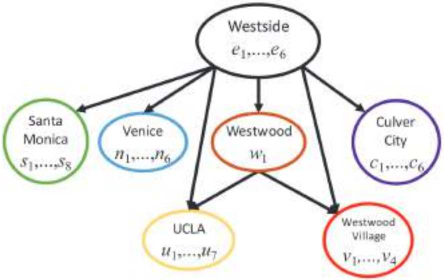
SOTA on modeling densities over rankings

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

SOP: Routing



Decomposing complex (conditional) probability spaces via circuits



Shen et al., "Conditional PSDDs: Modeling and learning with modular knowledge", 2018
Shen et al., "Structured Bayesian Networks: From Inference to Learning with Routes", 2019

Challenge #3

scaling tractable learning

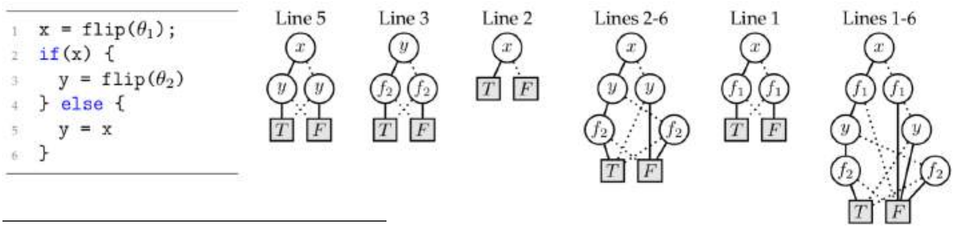
Learn tractable models

on **millions of datapoints**

and **thousands of features**

in tractable time!

Probabilistic programming



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...

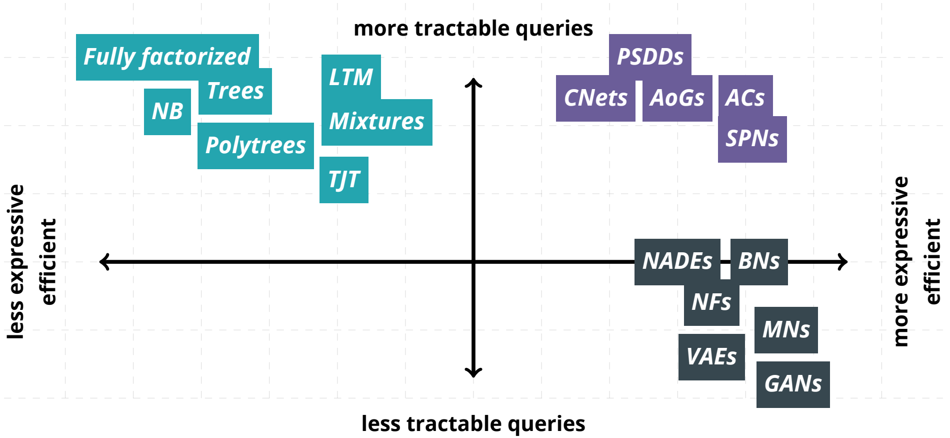
fault prediction *[Nath et al. 2016]*
computational psychology *[Joshi et al. 2018]*
biology *[Butz et al. 2018b]*
low-energy prediction *[Galindez Olascoaga et al. 2019; Shah et al. 2019]*
calibration of analog/RF circuits *[Andraud et al. 2018]*
stochastic constraint optimization *[Latour et al. 2017]*
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]*

Challenge #4

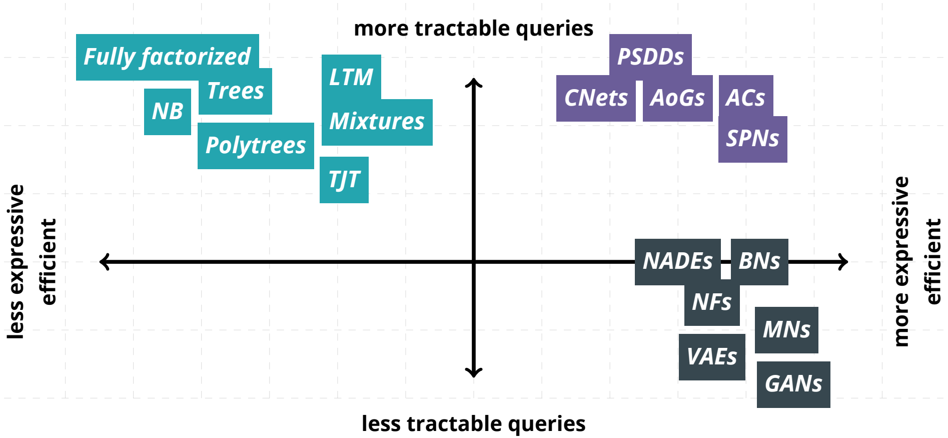
better benchmarks

Move beyond toy queries

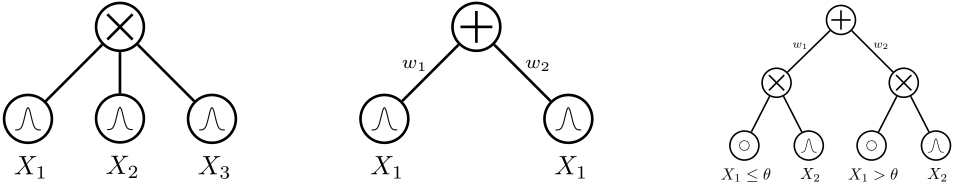
towards **fully automated reasoning**!



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***

Open challenges

1. new benchmarks are needed!
2. scaling tractable learning!
3. take the best from approximate reasoning!
4. move to complex reasoning!

takeaway #4: ***lots to do still,...***

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.
- Darwiche, Adnan (2009). *Modeling and Reasoning with Bayesian Networks*. Cambridge.
- Koller, Daphne and Nir Friedman (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.
- Choi, Arthur and Adnan Darwiche (2010). “Relax, compensate and then recover”. In: *JSAI International Symposium on Artificial Intelligence*. Springer, pp. 167–180.
- 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.
- 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.
- Domingos, Pedro and William Austin Webb (2012). “A tractable first-order probabilistic logic”. In: *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

References IV

- Gens, Robert and Pedro Domingos (2012). “Discriminative Learning of Sum-Product Networks”. In: *Advances in Neural Information Processing Systems 25*, pp. 3239–3247.
- 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.
- Lee, Sang-Woo, Min-Oh Heo, and Byoung-Tak Zhang (2013). “Online Incremental Structure Learning of Sum-Product Networks”. In: *Neural Information Processing: 20th International Conference, ICONIP 2013, Daegu, Korea, November 3-7, 2013. Proceedings, Part II*. Ed. by Minho Lee et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 220–227. ISBN: 978-3-642-42042-9. DOI: `10.1007/978-3-642-42042-9_28`. URL: `http://dx.doi.org/10.1007/978-3-642-42042-9_28`.
- Liu, Qiang and Alexander Ihler (2013). “Variational algorithms for marginal MAP”. In: *The Journal of Machine Learning Research* 14.1, pp. 3165–3200.
- 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.

References V

- 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`.
- Lee, Sang-Woo, Christopher Watkins, and Byoung-Tak Zhang (2014). “Non-Parametric Bayesian Sum-Product Networks”. In: *Workshop on Learning Tractable Probabilistic Models*. Citeseer.
- 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*.
- 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.

References VI

- 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*.
- 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*.
- 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)*.
- 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.

References VII

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

References VIII

- 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`.
- 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]`.
- Rahman, Tahrima and Vibhav Gogate (2016a). "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`.
- — (2016b). "Merging Strategies for Sum-Product Networks: From Trees to Graphs". In: *UAI*, ??–??

References IX

- 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.
- Sguerra, Bruno Massoni and Fabio G Cozman (2016). “Image classification using sum-product networks for autonomous flight of micro aerial vehicles”. In: *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*. IEEE, pp. 139–144.
- Shen, Yujia, Arthur Choi, and Adnan Darwiche (2016). “Tractable Operations for Arithmetic Circuits of Probabilistic Models”. In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3936–3944.
- Vlasselaer, Jonas 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.
- 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*.

References X

- 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*.
- Hsu, Wilson, Agastya Kalra, and Pascal Poupart (2017). “Online Structure Learning for Sum-Product Networks with Gaussian Leaves”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. URL: `https://openreview.net/forum?id=By7LxZNFe`.
- Kimmig, Angelika, Guy Van den Broeck, and Luc De Raedt (2017). “Algebraic model counting”. In: *Journal of Applied Logic* 22, pp. 46–62.
- 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`.
- 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.

References XI

- 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.
- 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. (2018a). "An Empirical Study of Methods for SPN Learning and Inference". In: *International Conference on Probabilistic Graphical Models*, pp. 49–60.
- Butz, Cory J et al. (2018b). "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*.
- 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.

References XII

- 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*.
- 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.
- 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*.
- Vergari, Antonio et al. (2018). “Automatic Bayesian Density Analysis”. In: *CoRR* abs/1807.09306. arXiv: `1807.09306`. URL: `http://arxiv.org/abs/1807.09306`.
- 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*.

References XIII

- Desana, Mattia and Christoph Schnörr (2019). "Sum–product graphical models". In: *Machine Learning*.
- 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`.
- 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)*.
- 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.
- 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*.
- 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`.

References XIV

- 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: *CoRR* abs/1905.10884. arXiv: 1905.10884. URL: http://arxiv.org/abs/1905.10884.