

Probabilistic Circuits

***Representations
Inference
Learning
Applications***

Antonio Vergari

University of California, Los Angeles

Robert Peharz

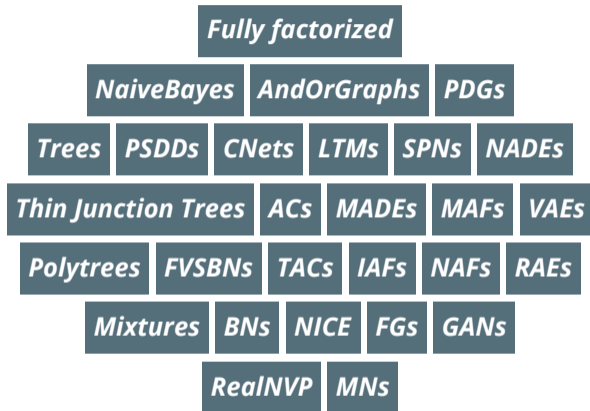
TU Eindhoven

Yoojung Choi

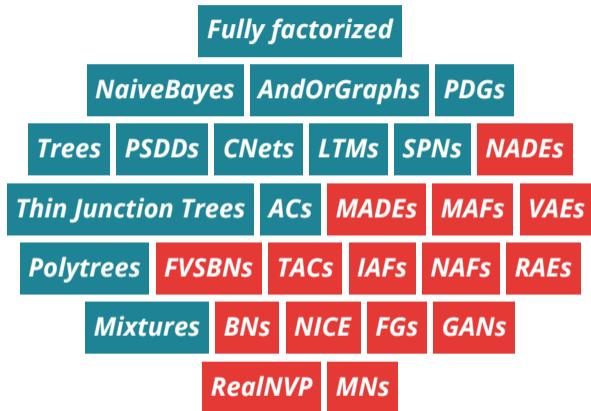
University of California, Los Angeles

Guy Van den Broeck

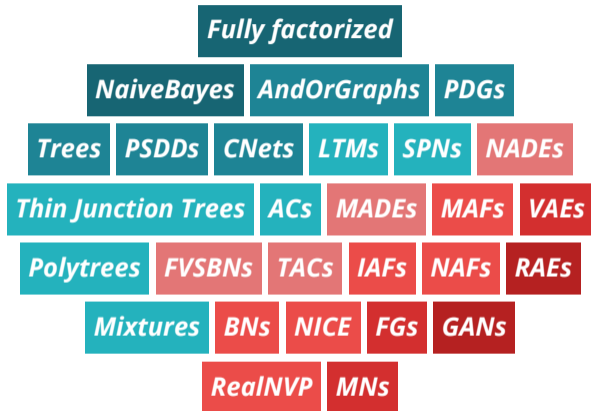
University of California, Los Angeles



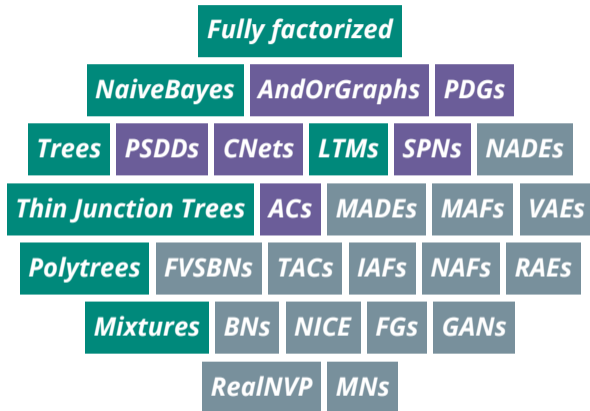
The Alphabet Soup of probabilistic models



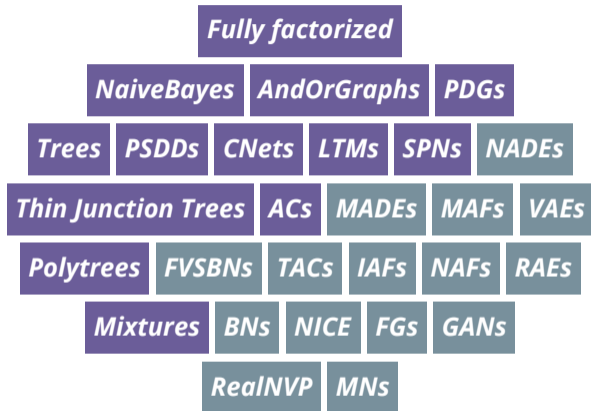
Intractable and ***tractable*** models



tractability is a spectrum



Expressive* models without *compromises



a *unifying framework* for tractable models

Why tractable inference?

or expressiveness vs tractability

Why tractable inference?

or expressiveness vs tractability

Probabilistic circuits

a unified framework for tractable probabilistic modeling

Why tractable inference?

or expressiveness vs tractability

Probabilistic circuits

a unified framework for tractable probabilistic modeling

Learning circuits

learning their structure and parameters from data

Why tractable inference?

or expressiveness vs tractability

Probabilistic circuits

a unified framework for tractable probabilistic modeling

Learning circuits

learning their structure and parameters from data

Representations and theory

tracing the boundaries of tractability and connections to other formalisms

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 5th Avenue?*



© fineartamerica.com

Why probabilistic inference?

- q₁**: *What is the probability that today is a Monday and there is a traffic jam on 5th Avenue?*
- q₂**: *Which day is most likely to have a traffic jam on my route to work?*



© fineartamerica.com

Why probabilistic inference?

q₁: *What is the probability that today is a Monday and there is a traffic jam on 5th Avenue?*

q₂: *Which day is most likely to have a traffic jam on my route to work?*

⇒ fitting a predictive model!



© fineartamerica.com

Why probabilistic inference?

q_1 : *What is the probability that today is a Monday and there is a traffic jam on 5th Avenue?*

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

⇒ ~~fitting a predictive model!~~



© fineartamerica.com

Why probabilistic inference?

q_1 : *What is the probability that today is a Monday and there is a traffic jam on 5th Avenue?*

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

⇒ ~~fitting a predictive model!~~

⇒ answering probabilistic **queries** on a probabilistic model of the world **m**

$$q_1(\mathbf{m}) = ?$$

$$q_2(\mathbf{m}) = ?$$



© fineartamerica.com

Why probabilistic inference?

q_1 : What is the probability that today is a Monday and there is a traffic jam on 5th Avenue?

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

$q_1(\mathbf{m}) = p_{\mathbf{m}}(\text{Day} = \text{Mon}, \text{Jam}_{5\text{th}} = 1)$



© fineartamerica.com

Why probabilistic inference?

q_1 : What is the probability that today is a Monday and there is a traffic jam on 5th Avenue?

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

$q_1(\mathbf{m}) = p_{\mathbf{m}}(\text{Day} = \text{Mon}, \text{Jam}_{5\text{th}} = 1)$

\Rightarrow *marginals*



© fineartamerica.com

Why probabilistic inference?

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

$\mathbf{X} = \{\text{Day}, \text{Time}, \text{Jam}_{\text{Str}1}, \text{Jam}_{\text{Str}2}, \dots, \text{Jam}_{\text{Str}N}\}$

$q_2(\mathbf{m}) = \operatorname{argmax}_d p_{\mathbf{m}}(\text{Day} = d \wedge \bigvee_{i \in \text{route}} \text{Jam}_{\text{Str}i})$



© fineartamerica.com

Why probabilistic inference?

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

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

$q_2(\mathbf{m}) = \operatorname{argmax}_d p_{\mathbf{m}}(\text{Day} = d \wedge \bigvee_{i \in \text{route}} \text{Jam}_{\text{Str}i})$

\Rightarrow *marginals + MAP + logical events*



© fineartamerica.com

Tractable Probabilistic Inference

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

Tractable Probabilistic Inference

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

\Rightarrow 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 $q \in \mathcal{Q}$ and model $m \in \mathcal{M}$ **exactly** computing $q(m)$ runs in time $O(\text{poly}(|m|))$.

\Rightarrow often poly will in fact be **linear!**

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

Why exact inference?

or “What about approximate inference?”

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

Why exact inference?

or “What about approximate inference?”

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

Why exact inference?

or “What about approximate inference?”

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

Why exact inference?

or “What about approximate inference?”

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

Why exact inference?

or “What about approximate inference?”

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

Stay tuned for...

Next:

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

After:

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

Complete evidence (EVI)

q₃: *What is the probability that today is a Monday at 12.00 and there is a traffic jam only on 5th Avenue?*



© fineartamerica.com

Complete evidence (EVI)

q_3 : What is the probability that today is a Monday at 12.00 and there is a traffic jam only on 5th Avenue?

$\mathbf{X} = \{\text{Day, Time, Jam}_{5\text{th}}, \text{Jam}_{\text{Str}2}, \dots, \text{Jam}_{\text{Str}N}\}$

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



© fineartamerica.com

Complete evidence (EVI)

q_3 : What is the probability that today is a Monday at 12.00 and there is a traffic jam only on 5th Avenue?

$\mathbf{X} = \{\text{Day, Time, Jam}_{5\text{th}}, \text{Jam}_{\text{Str}2}, \dots, \text{Jam}_{\text{Str}N}\}$

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

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

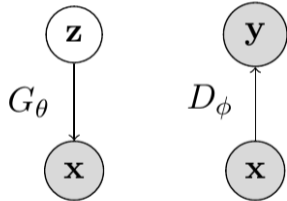
$$\theta_{\mathbf{m}}^{\text{MLE}} = \operatorname{argmax}_{\theta} \prod_{\mathbf{x} \in \mathcal{D}} p_{\mathbf{m}}(\mathbf{x}; \theta)$$



© fineartamerica.com

Generative Adversarial Networks

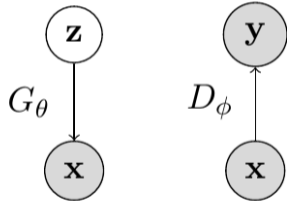
$$\min_{\theta} \max_{\phi} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D_{\phi}(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [\log(1 - D_{\phi}(G_{\theta}(\mathbf{z})))]$$



Generative Adversarial Networks

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

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



M

Q:

EVI

GANs

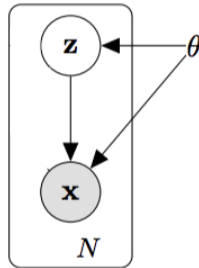


tractable bands

Variational Autoencoders

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

- an explicit likelihood model!

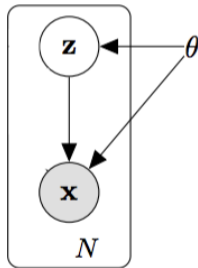


Rezende et al., "Stochastic backprop. and approximate inference in deep generative models", 2014
Kingma et al., "Auto-Encoding Variational Bayes", 2014

Variational Autoencoders

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

- an explicit likelihood model!
- ... but computing $\log p_{\theta}(\mathbf{x})$ is intractable
 - \Rightarrow an infinite and uncountable mixture
 - \Rightarrow no tractable EVI
- we need to optimize the ELBO...
 - \Rightarrow which is “tricky” [Alemi et al. 2017; Dai et al. 2019; Ghosh et al. 2019]



\mathcal{M} \mathcal{Q} :

EVI

GANs

X

VAEs

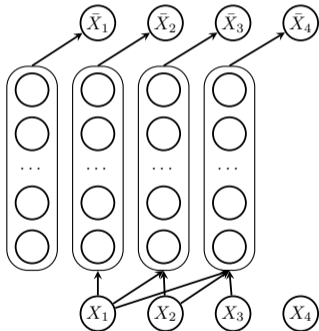
X

tractable bands

Autoregressive models

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

- an explicit likelihood!
- ...as a product of factors \Rightarrow tractable EVI!
- many neural variants
 - NADE [Larochelle et al. 2011],
MADE [Germain et al. 2015]
 - PixelCNN [Salimans et al. 2017],
PixelRNN [Oord et al. 2016]



Marginal queries (MAR)

q_1 : What is the probability that today is a Monday ~~at~~
~~12:00~~ and there is a traffic jam ~~only~~ on 5th
Avenue?



© fineartamerica.com

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 5th
Avenue?

$$q_1(\mathbf{m}) = p_{\mathbf{m}}(\text{Day} = \text{Mon}, \text{Jam}_{5\text{th}} = 1)$$



© fineartamerica.com

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 5th
Avenue?

$$q_1(\mathbf{m}) = p_{\mathbf{m}}(\text{Day} = \text{Mon}, \text{Jam}_{5\text{th}} = 1)$$

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

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



© fineartamerica.com

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 5th
Avenue?

$$q_1(\mathbf{m}) = p_{\mathbf{m}}(\text{Day} = \text{Mon}, \text{Jam}_{5\text{th}} = 1)$$

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

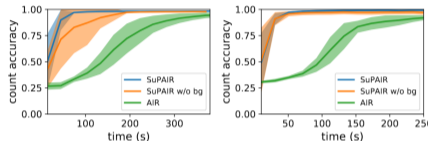
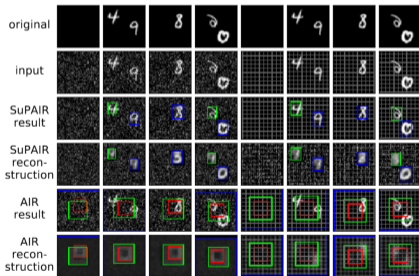
and if you can answer MAR queries,
then you can also do **conditional queries** (CON):

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



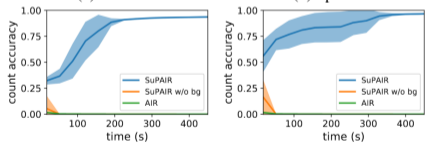
© fineartamerica.com

Tractable MAR : scene understanding



(a) MNIST

(b) Sprites



(c) Noisy MNIST

(d) Grid MNIST

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

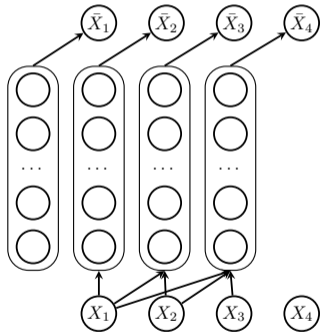
Stelzner et al., “Faster Attend-Infer-Repeat with Tractable Probabilistic Models”, 2019

Kossen et al., “Structured Object-Aware Physics Prediction for Video Modeling and Planning”, 2019

Autoregressive models

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

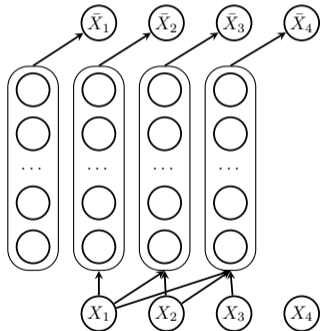
- an explicit likelihood!
- ...as a product of factors \Rightarrow tractable EVI!



Autoregressive models

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

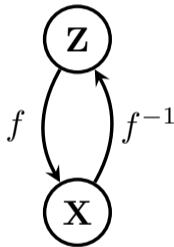
- an explicit likelihood!
- ...as a product of factors \Rightarrow tractable EVI!
- ... but we need to fix a variable ordering
 \Rightarrow **only some** MAR queries are tractable
for one ordering



Normalizing flows

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

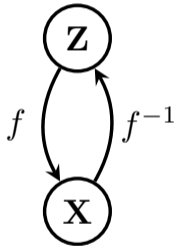
- an explicit likelihood \Rightarrow tractable EVI!
- ... computing the determinant of the Jacobian

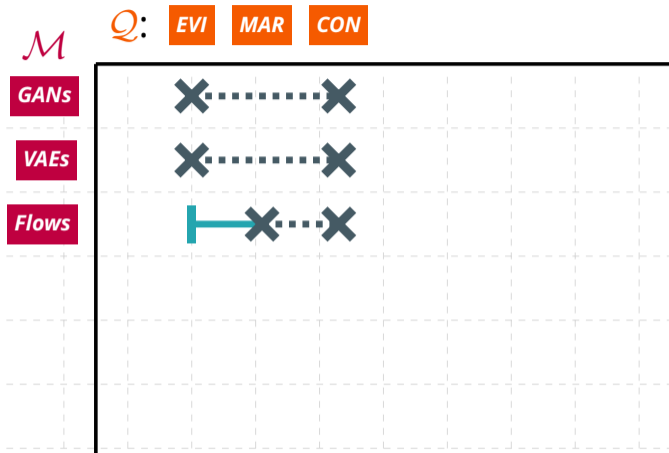


Normalizing flows

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

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





tractable bands

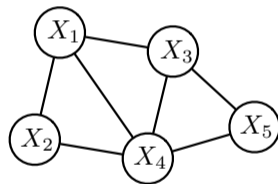
Probabilistic Graphical Models (PGMs)

Declarative semantics: a clean separation of modeling assumptions from inference

Nodes: random variables

Edges: dependencies

+



Inference:

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

Complexity of MAR on PGMs

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

Why? Treewidth!

Treewidth:

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

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

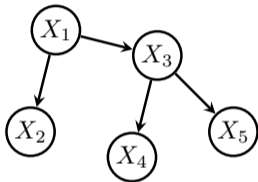
Fixed-parameter tractable: MAR and CON on a graphical model \mathbf{m} with treewidth w take time $O(|\mathbf{X}| \cdot 2^w)$, which is linear for fixed width w

[Dechter 1998; Koller et al. 2009].



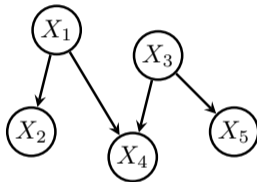
what about bounding the treewidth by design?

Low-treewidth PGMs



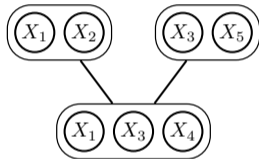
Trees

[Meilă et al. 2000]



Polytrees

[Dasgupta 1999]



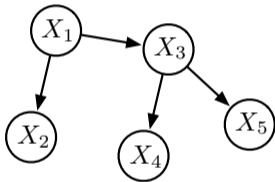
Thin Junction trees

[Bach et al. 2001]

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

Tree distributions

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

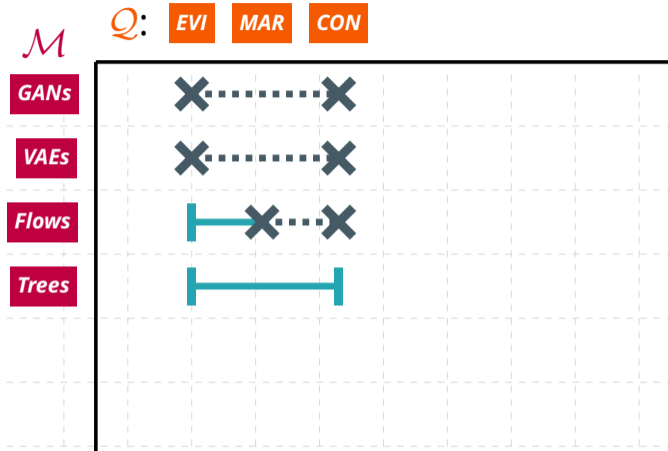


$$p(\mathbf{X}) = \prod_{i=1}^n p(x_i | \text{Pa}_{x_i})$$

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 algorithm¹

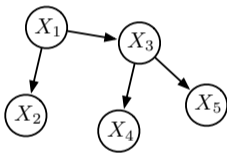
¹Chow et al., "Approximating discrete probability distributions with dependence trees", 1968



tractable bands

What do we lose?

Expressiveness: Ability to represent rich and complex classes of distributions

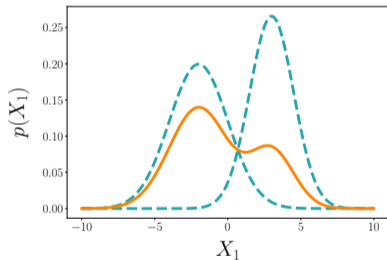


Bounded-treewidth PGMs lose the ability to represent *all possible distributions* ...

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

Mixtures

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

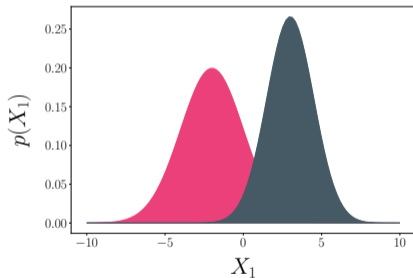


$$p(X) = w_1 \cdot p_1(X) + w_2 \cdot p_2(X)$$

EVI, MAR, CON queries scale linearly in k

Mixtures

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



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

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

\Rightarrow increased expressiveness

Expressiveness and efficiency

Expressiveness: Ability to represent rich and effective classes of functions

⇒ *mixture of Gaussians can approximate any distribution!*

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

Expressiveness and efficiency

Expressiveness: Ability to represent rich and effective classes of functions

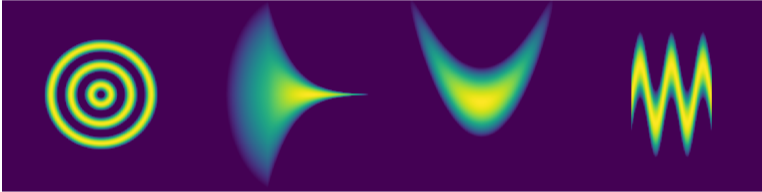
⇒ *mixture of Gaussians can approximate any distribution!*

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

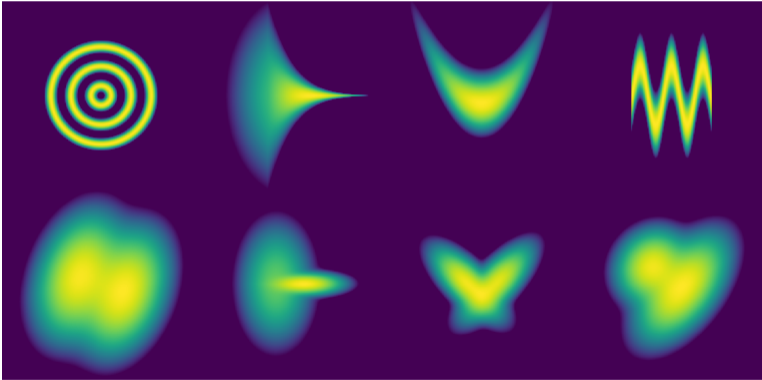
⇒ *but how many components does a Gaussian mixture need?*

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

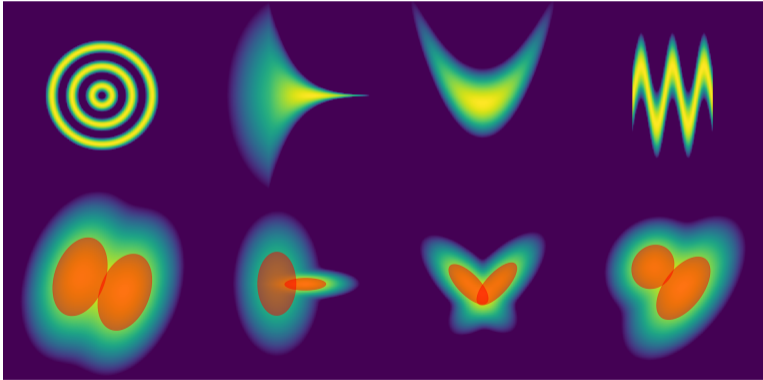
How expressive efficient are mixture?



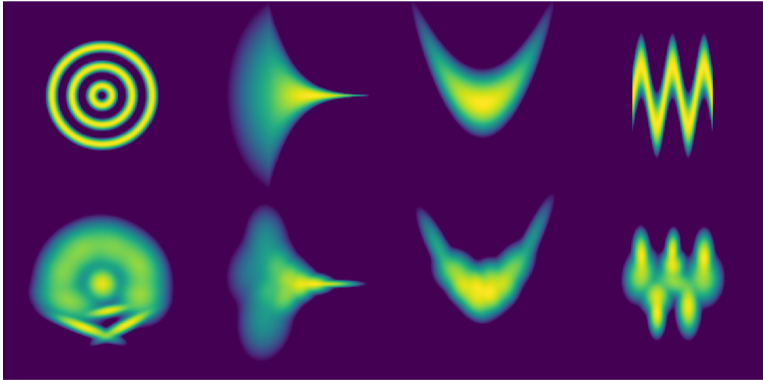
How expressive efficient are mixture?



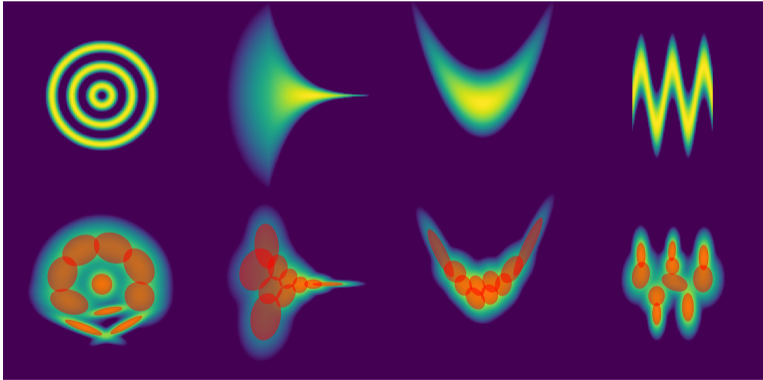
How expressive efficient are mixture?



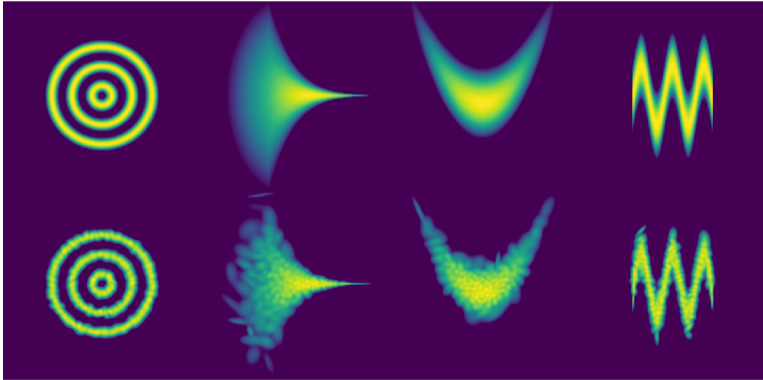
How expressive efficient are mixture?



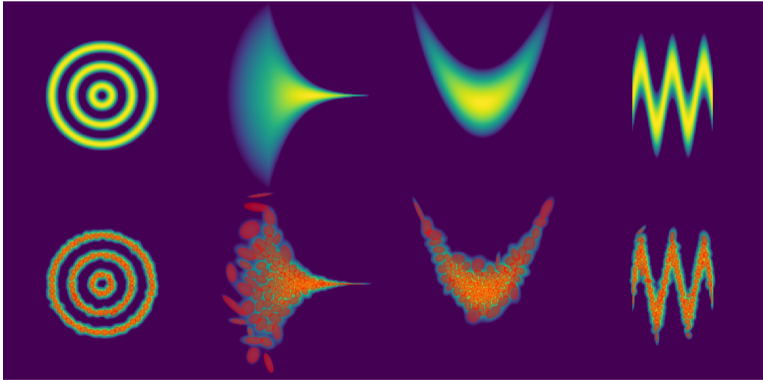
How expressive efficient are mixture?



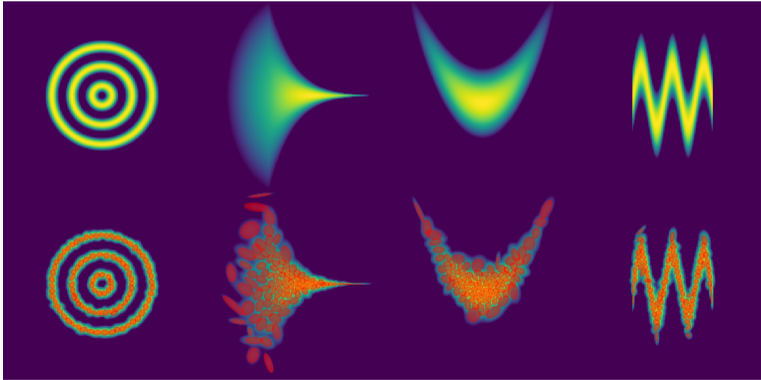
How expressive efficient are mixture?



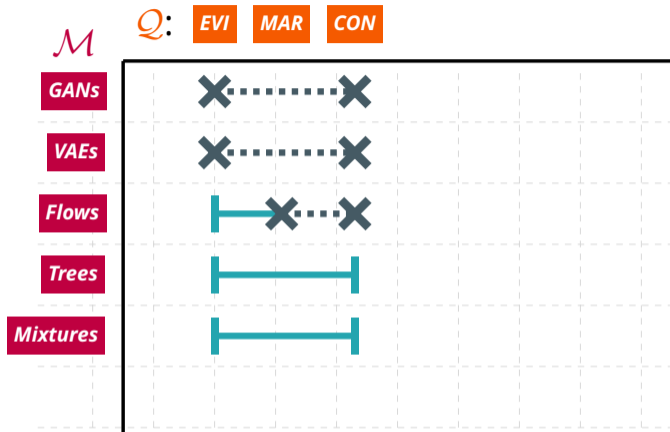
How expressive efficient are mixture?



How expressive efficient are mixture?



⇒ *stack mixtures like in deep generative models* ^{37/158}



tractable bands

Maximum A Posteriori (MAP)

aka Most Probable Explanation (MPE)

Q₅: *Which combination of roads is most likely to be jammed on Monday at 9am?*



© fineartamerica.com

Maximum A Posteriori (MAP)

aka Most Probable Explanation (MPE)

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

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



© fineartamerica.com

Maximum A Posteriori (MAP)

aka Most Probable Explanation (MPE)

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

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

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

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



© fineartamerica.com

Maximum A Posteriori (MAP)

aka Most Probable Explanation (MPE)

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

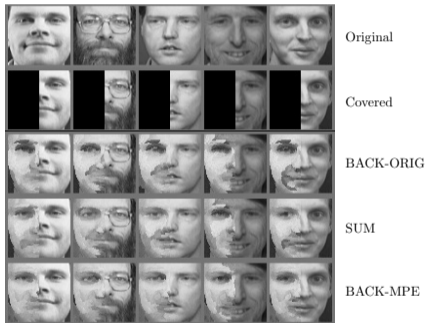
...**intractable** for latent variable models!

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



© fineartamerica.com

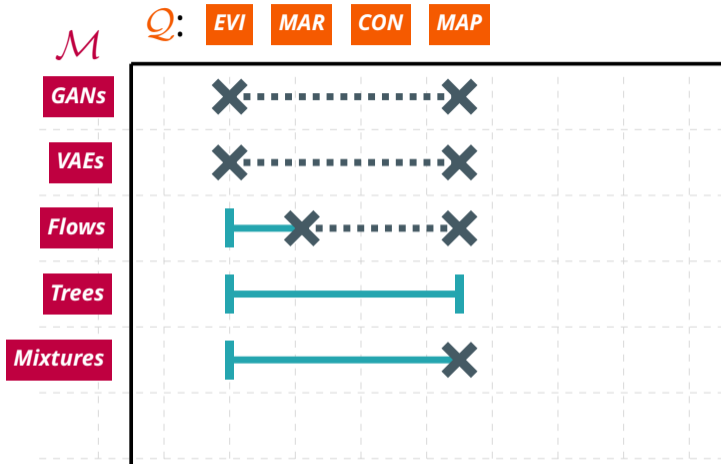
MAP inference : image inpainting



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

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



tractable bands

Marginal MAP (MMAP)

aka Bayesian Network MAP

Q₆: Which combination of roads is most likely to be jammed ~~on Monday~~ at 9am?



© fineartamerica.com

Marginal MAP (MMAP)

aka Bayesian Network MAP

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

$$q_6(\mathbf{m}) = \operatorname{argmax}_{\mathbf{j}} p_{\mathbf{m}}(\mathbf{j}_1, \mathbf{j}_2, \dots \mid \text{Time}=9)$$



© fineartamerica.com

Marginal MAP (MMAP)

aka Bayesian Network MAP

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

$$q_6(\mathbf{m}) = \operatorname{argmax}_{\mathbf{j}} p_{\mathbf{m}}(\mathbf{j}_1, \mathbf{j}_2, \dots \mid \text{Time}=9)$$

$$\begin{aligned} \text{General: } & \operatorname{argmax}_{\mathbf{q}} p_{\mathbf{m}}(\mathbf{q} \mid \mathbf{e}) \\ & = \operatorname{argmax}_{\mathbf{q}} \sum_{\mathbf{h}} p_{\mathbf{m}}(\mathbf{q}, \mathbf{h} \mid \mathbf{e}) \end{aligned}$$

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



© fineartamerica.com

Marginal MAP (MMAP)

aka Bayesian Network MAP

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

$$q_6(\mathbf{m}) = \operatorname{argmax}_{\mathbf{j}} p_{\mathbf{m}}(\mathbf{j}_1, \mathbf{j}_2, \dots \mid \text{Time}=9)$$

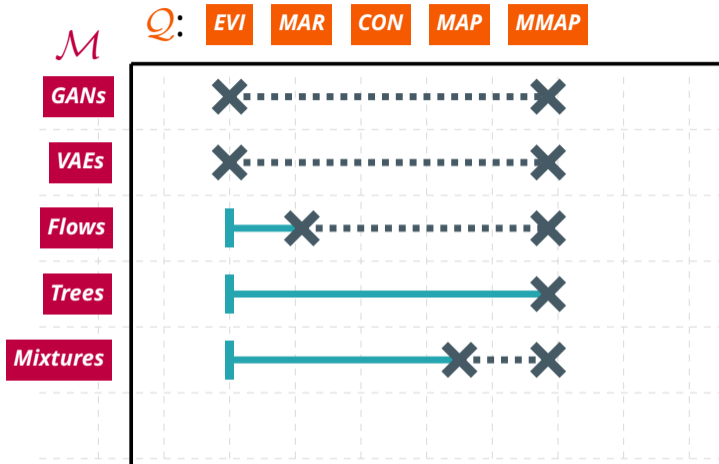
\Rightarrow NP^{PP} -complete [Park et al. 2006]

\Rightarrow NP-hard for trees [Campos 2011]

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



© fineartamerica.com



tractable bands

Advanced queries

q₂: Which day is most likely to have a traffic jam on my route to work?



© fineartamerica.com

Advanced queries

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

$$q_2(\mathbf{m}) = \operatorname{argmax}_d p_{\mathbf{m}}(\text{Day} = d \wedge \bigvee_{i \in \text{route}} \text{Jam}_{\text{Str } i})$$

\Rightarrow *marginals + MAP + logical events*



© fineartamerica.com

Advanced queries

- q₂**: Which day is most likely to have a traffic jam on my route to work?
- q₇**: What is the probability of seeing more traffic jams in Uptown than Midtown?



© fineartamerica.com

Advanced queries

q₂: Which day is most likely to have a traffic jam on my route to work?

q₇: What is the probability of seeing more traffic jams in Uptown than Midtown?

⇒ **counts + group comparison**



© fineartamerica.com

Advanced queries

q₂: Which day is most likely to have a traffic jam on my route to work?

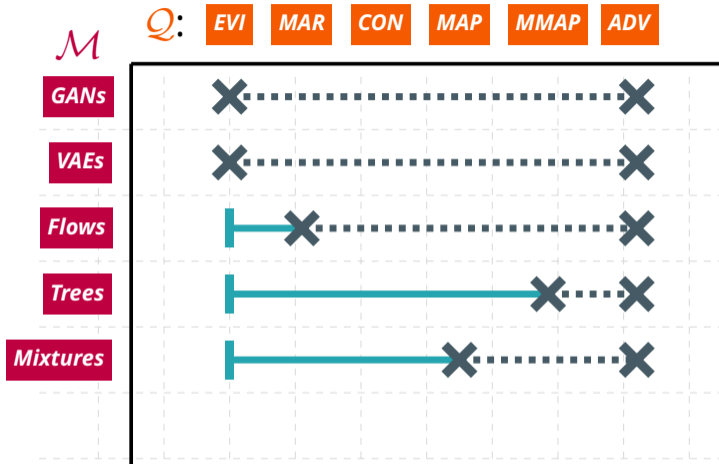
q₇: What is the probability of seeing more traffic jams in Uptown than Midtown?

and more:

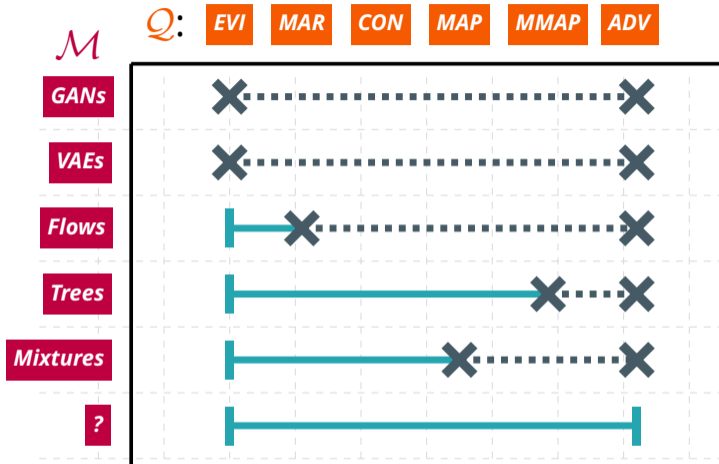
- expected classification agreement
[Oztok et al. 2016; Choi et al. 2017, 2018]
- expected predictions *[Khosravi et al. 2019b]*



© fineartamerica.com



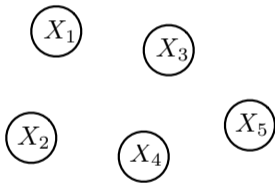
tractable bands



tractable bands

Fully factorized models

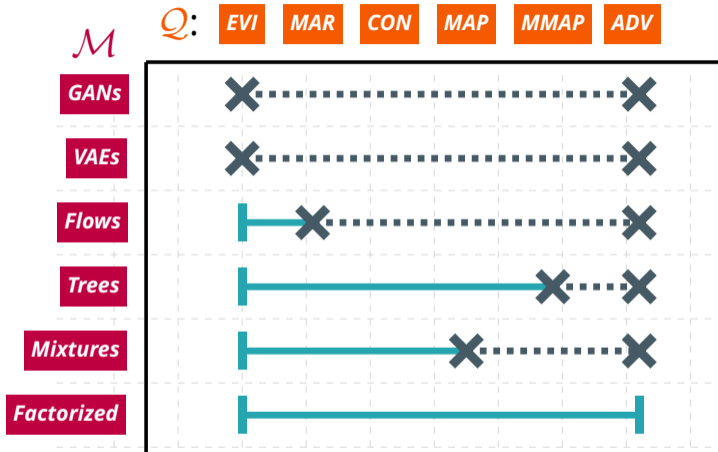
A completely disconnected graph. Example: Product of Bernoullis (PoBs)



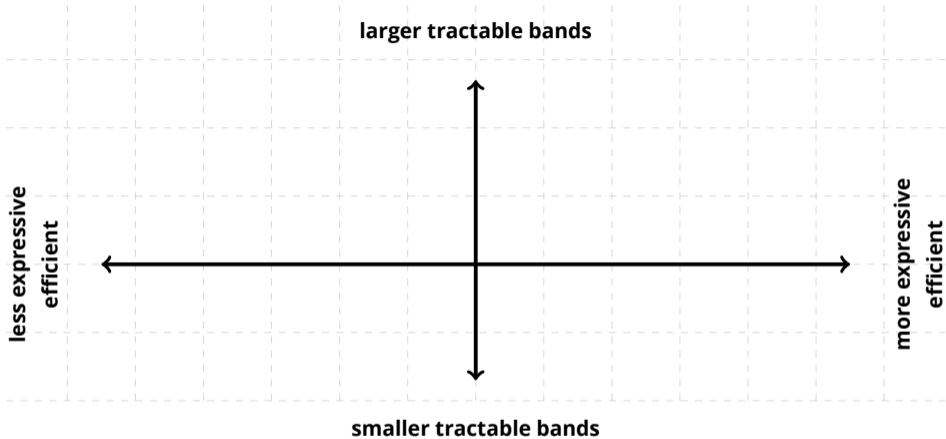
$$p(\mathbf{x}) = \prod_{i=1}^n p(x_i)$$

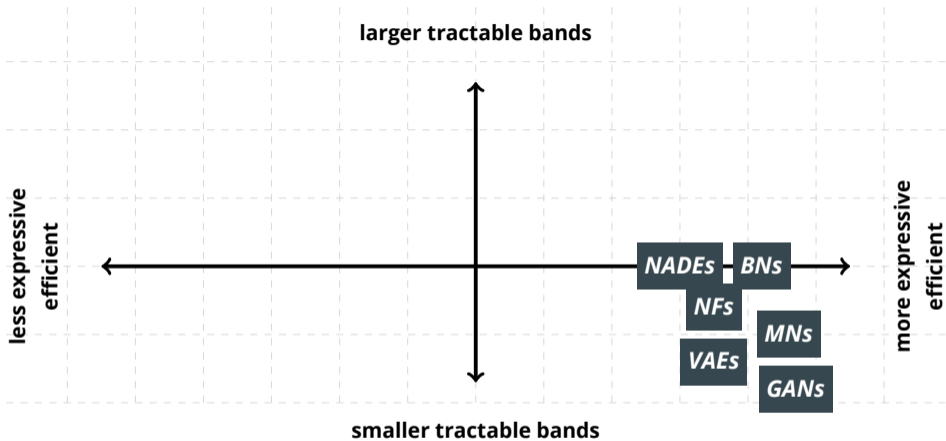
Complete evidence, marginals and MAP, MMAP inference is **linear**!

⇒ *but definitely not expressive...*

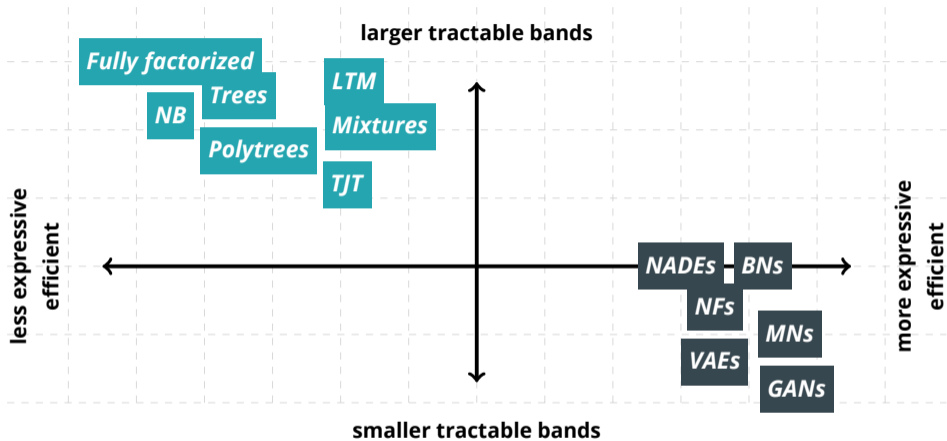


tractable bands

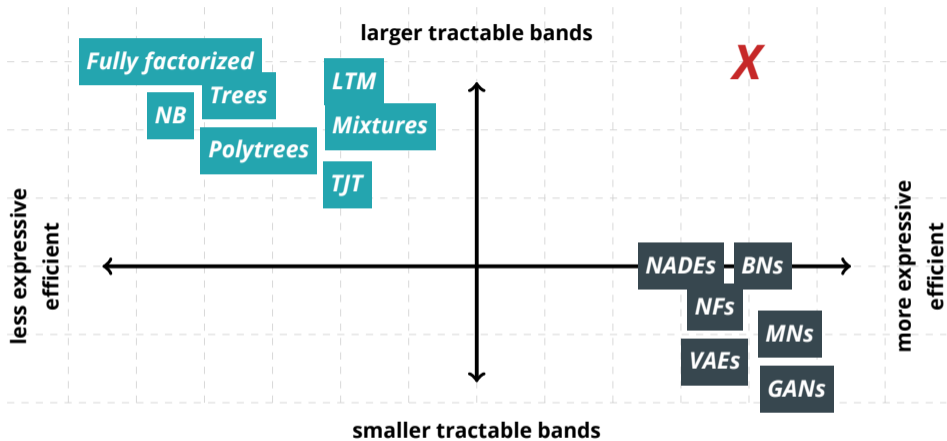




Expressive models are not very tractable...



and tractable ones are not very expressive...



probabilistic circuits are at the "sweet spot"

Probabilistic Circuits

Probabilistic circuits

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

Probabilistic circuits

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

\Rightarrow operational semantics!

Probabilistic circuits

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

\Rightarrow operational semantics!

\Rightarrow by constraining the graph we can make inference tractable...

Stay tuned for...

Next:

1. *What are the building blocks of probabilistic circuits?*
⇒ *How to build a tractable computational graph?*
2. *For which queries are probabilistic circuits tractable?*
⇒ *tractable classes induced by structural properties*

After:

How can probabilistic circuits be learned?

Distributions as computational graphs



Base case: a single node encoding a distribution

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

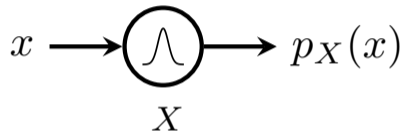
Distributions as computational graphs



Base case: a single node encoding a distribution

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

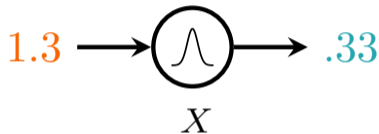
Distributions as computational graphs



Simple distributions are tractable “black boxes” for:

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

Distributions as computational graphs



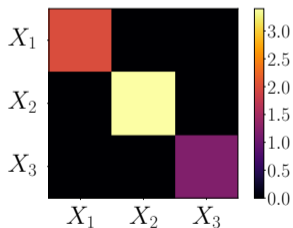
Simple distributions are tractable “black boxes” for:

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

Factorizations as product nodes

Divide and conquer complexity

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

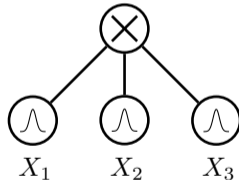
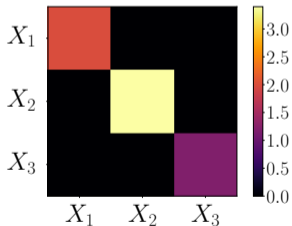


⇒ e.g. modeling a multivariate Gaussian with diagonal covariance matrix...

Factorizations as product nodes

Divide and conquer complexity

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

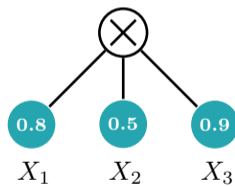
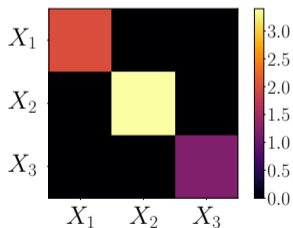


\Rightarrow ...with a product node over some univariate Gaussian distribution

Factorizations as product nodes

Divide and conquer complexity

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

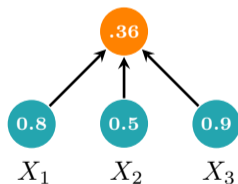
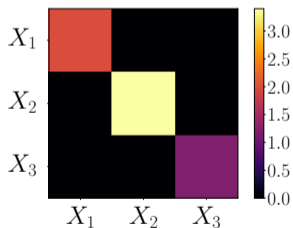


\Rightarrow *feedforward evaluation*

Factorizations as product nodes

Divide and conquer complexity

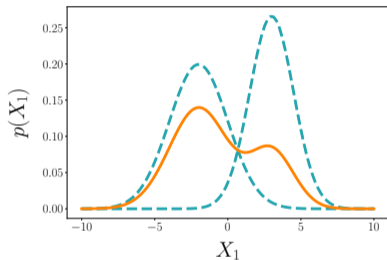
$$p(x_1, x_2, x_3) = p(x_1) \cdot p(x_2) \cdot p(x_3)$$



\Rightarrow *feedforward evaluation*

Mixtures as sum nodes

Enhance expressiveness

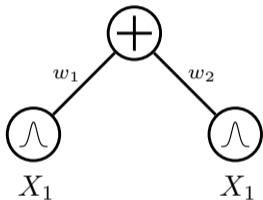


$$p(X) = w_1 \cdot p_1(X) + w_2 \cdot p_2(X)$$

⇒ e.g. modeling a mixture of Gaussians...

Mixtures as sum nodes

Enhance expressiveness

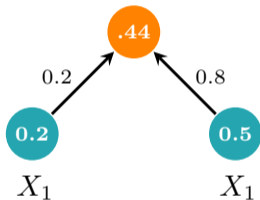


$$p(x) = 0.2 \cdot p_1(x) + 0.8 \cdot p_2(x)$$

⇒ ...as weighted a sum node over Gaussian input distributions

Mixtures as sum nodes

Enhance expressiveness



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

\Rightarrow by **stacking** them we increase expressive efficiency

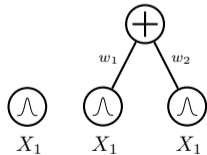
A grammar for tractable models

Recursive semantics of probabilistic circuits


 X_1

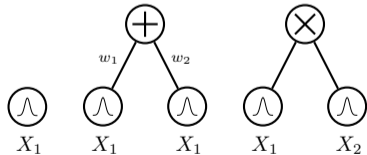
A grammar for tractable models

Recursive semantics of probabilistic circuits



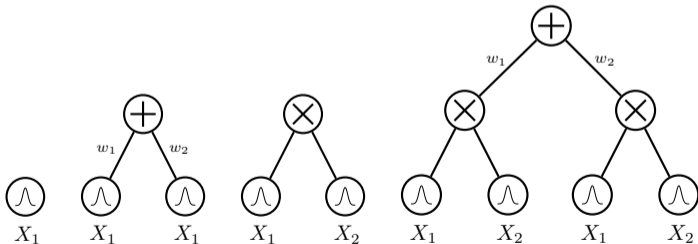
A grammar for tractable models

Recursive semantics of probabilistic circuits



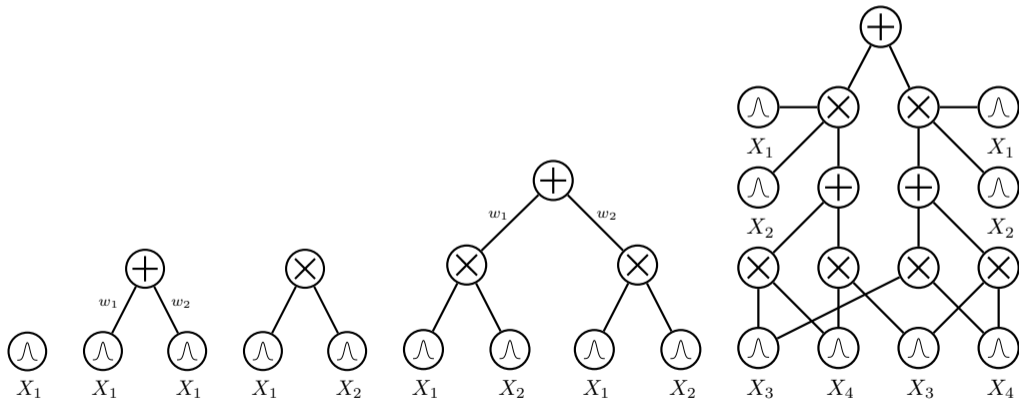
A grammar for tractable models

Recursive semantics of probabilistic circuits



A grammar for tractable models

Recursive semantics of probabilistic circuits



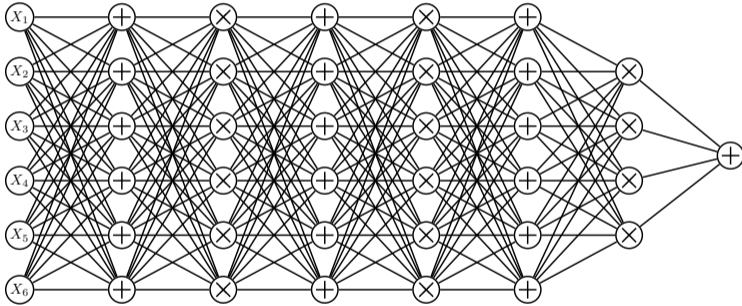
Probabilistic circuits are not PGMs!

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

	PGMs	Circuits
Nodes:	random variables	unit of computations
Edges:	dependencies	order of execution
Inference:	<ul style="list-style-type: none">■ conditioning■ elimination■ message passing	<ul style="list-style-type: none">■ feedforward pass■ backward pass

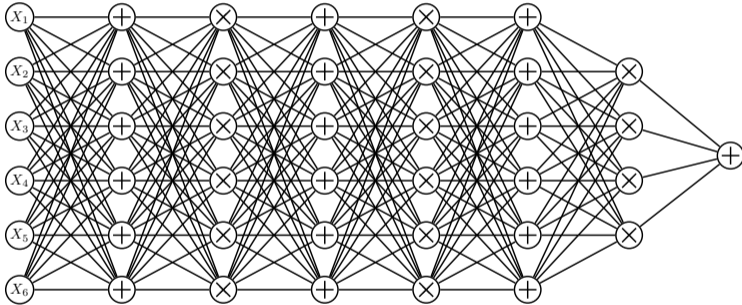
⇒ they are **computational graphs**, more like neural networks

Just sum, products and distributions?



just arbitrarily compose them like a neural network!

Just sum, products and distributions?



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



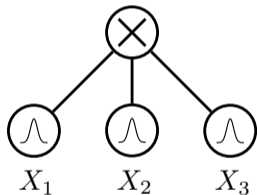
structural constraints needed for tractability

***Which structural constraints
to ensure tractability?***

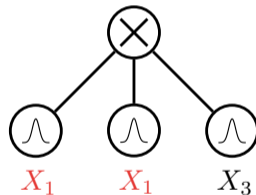
Decomposability

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

\Rightarrow just like in factorization!



decomposable circuit



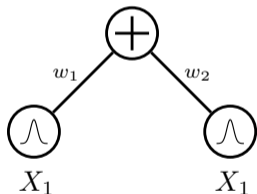
non-decomposable circuit

Smoothness

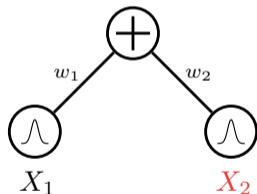
aka completeness

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

⇒ otherwise not accounting for some variables



smooth circuit



non-smooth circuit

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

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

Computing arbitrary integrations (or summations)

\Rightarrow *linear in circuit size!*

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

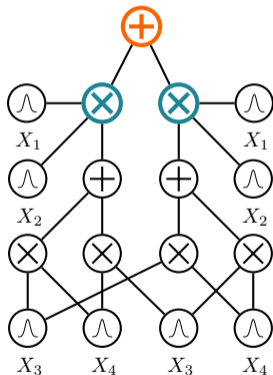
$$\int p(\mathbf{x}) d\mathbf{x}$$

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

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

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

\Rightarrow integrals are "pushed down" to children

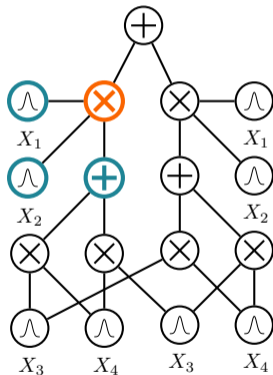


Smoothness + **decomposability** = **tractable MAR**

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

$$\begin{aligned} & \int \int \int p(\mathbf{x}, \mathbf{y}, \mathbf{z}) dx dy dz = \\ &= \int \int \int p(\mathbf{x})p(\mathbf{y})p(\mathbf{z}) dx dy dz = \\ &= \int p(\mathbf{x}) dx \int p(\mathbf{y}) dy \int p(\mathbf{z}) dz \end{aligned}$$

\Rightarrow integrals decompose into easier ones



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

Forward pass evaluation for MAR

\Rightarrow linear in circuit size!

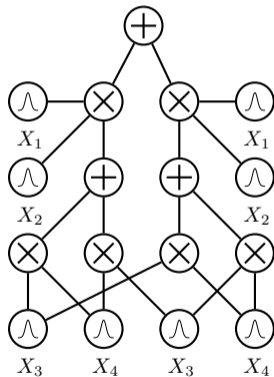
E.g. to compute $p(x_2, x_4)$:

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

\Rightarrow for normalized leaf distributions: 1.0

■ leafs over X_2 and X_4 output **EVI**

■ feedforward evaluation (bottom-up)



Smoothness + decomposability = tractable MAR

Forward pass evaluation for MAR

⇒ linear in circuit size!

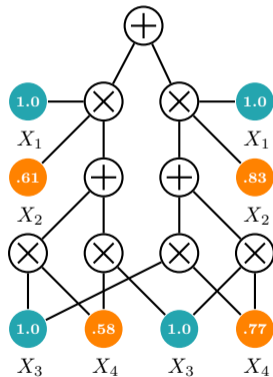
E.g. to compute $p(x_2, x_4)$:

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

⇒ for normalized leaf distributions: 1.0

leaves over X_2 and X_4 output **EVI**

feedforward evaluation (bottom-up)



Smoothness + decomposability = tractable MAR

Forward pass evaluation for MAR

⇒ linear in circuit size!

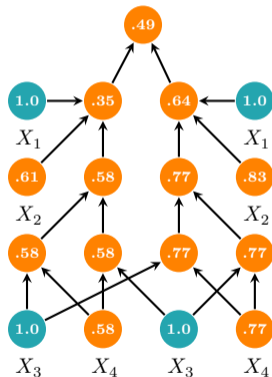
E.g. to compute $p(x_2, x_4)$:

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

⇒ for normalized leaf distributions: **1.0**

■ leafs over X_2 and X_4 output **EVI**

■ feedforward evaluation (bottom-up)

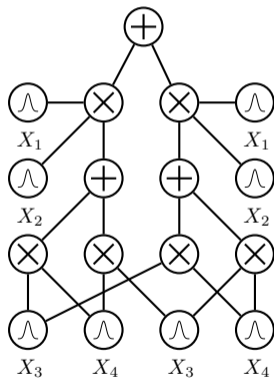


Smoothness + decomposability = tractable CON

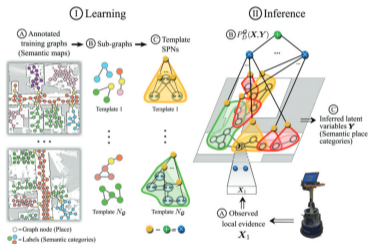
Analogously, for arbitrary conditional queries:

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

1. evaluate $p(\mathbf{q}, \mathbf{e}) \Rightarrow$ *one feedforward pass*
2. evaluate $p(\mathbf{e}) \Rightarrow$ *another feedforward pass*
 \Rightarrow *...still linear in circuit size!*



Tractable MAR : Robotics



Pixels for scenes and abstractions for maps decompose along circuit structures.

Fast and exact **marginalization** over unseen or “do not care” scene and map parts for **hierarchical planning robot executions**

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

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

We can also decompose bottom-up a MAP query:

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

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

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

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

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

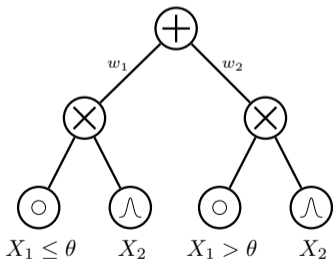
$$\operatorname{argmax}_{\mathbf{q}} \sum_i w_i p_i(\mathbf{q}, \mathbf{e}) = \operatorname{argmax}_{\mathbf{q}} \sum_{\mathbf{z}} p(\mathbf{q}, \mathbf{z}, \mathbf{e}) \neq \sum_{\mathbf{z}} \operatorname{argmax}_{\mathbf{q}} p(\mathbf{q}, \mathbf{z}, \mathbf{e})$$

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

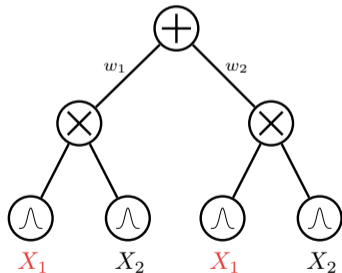
Determinism

aka selectivity

A sum node is deterministic if the output of only one children is non zero for any input
 \Rightarrow e.g. if their distributions have disjoint support



deterministic circuit



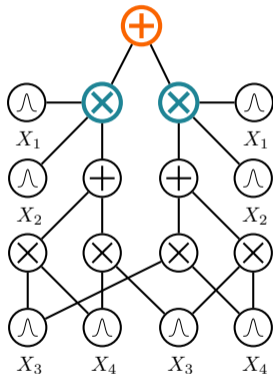
non-deterministic circuit

Determinism + decomposability = tractable MAP

If $p(\mathbf{q}, \mathbf{e}) = \sum_i w_i p_i(\mathbf{q}, \mathbf{e}) = \max_i w_i p_i(\mathbf{q}, \mathbf{e})$,
 (**deterministic** sum node):

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

⇒ *one non-zero child term, thus sum is max*

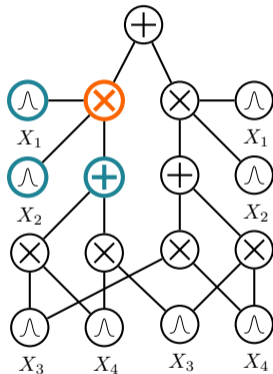


Determinism + **decomposability** = **tractable MAP**

If $p(\mathbf{q}, \mathbf{e}) = p(\mathbf{q}_x, \mathbf{e}_x, \mathbf{q}_y, \mathbf{e}_y) = p(\mathbf{q}_x, \mathbf{e}_x)p(\mathbf{q}_y, \mathbf{e}_y)$
 (**decomposable** product node):

$$\begin{aligned} \max_{\mathbf{q}} p(\mathbf{q} | \mathbf{e}) &= \max_{\mathbf{q}} p(\mathbf{q}, \mathbf{e}) \\ &= \max_{\mathbf{q}_x, \mathbf{q}_y} p(\mathbf{q}_x, \mathbf{e}_x, \mathbf{q}_y, \mathbf{e}_y) \\ &= \max_{\mathbf{q}_x} p(\mathbf{q}_x, \mathbf{e}_x), \max_{\mathbf{q}_y} p(\mathbf{q}_y, \mathbf{e}_y) \end{aligned}$$

\Rightarrow solving optimization independently



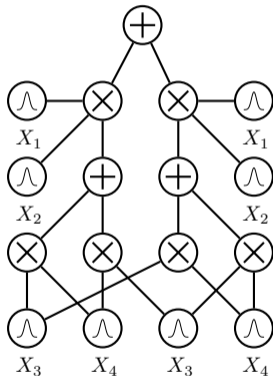
Determinism + **decomposability** = **tractable MAP**

Evaluating the circuit twice:

bottom-up and **top-down**



still linear in circuit size!



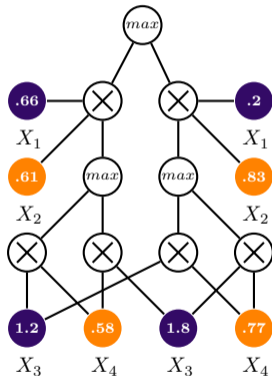
Determinism + decomposability = tractable MAP

Evaluating the circuit twice:

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

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

1. turn sum into max nodes and distributions into max distributions
2. evaluate $p(x_2, x_4)$ bottom-up
3. retrieve max activations top-down
4. compute **MAP states** for X_1 and X_3 at leaves



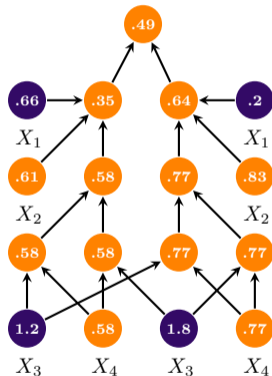
Determinism + decomposability = tractable MAP

Evaluating the circuit twice:

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

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

1. turn sum into max nodes and distributions into max distributions
2. evaluate $p(x_2, x_4)$ bottom-up
3. retrieve max activations top-down
4. compute **MAP states** for X_1 and X_3 at leaves



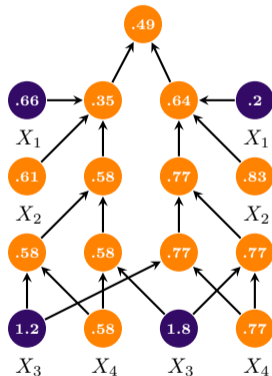
Determinism + **decomposability** = **tractable MAP**

Evaluating the circuit twice:

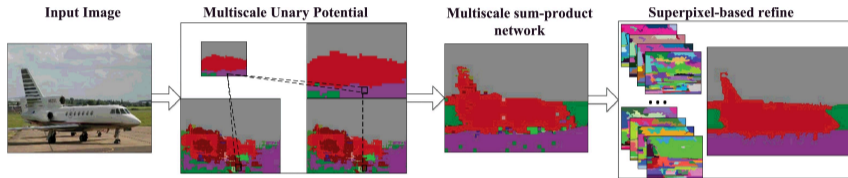
bottom-up and **top-down** \Rightarrow *still linear in circuit size!*

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

1. turn sum into max nodes and distributions into max distributions
2. evaluate $p(x_2, x_4)$ bottom-up
3. retrieve max activations top-down
4. compute **MAP states** for X_1 and X_3 at leaves



MAP inference : image segmentation



Semantic segmentation is MAP over joint pixel and label space

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

Rathke et al., "Locally adaptive probabilistic models for global segmentation of pathological oct scans", 2017

Yuan et al., "Modeling spatial layout for scene image understanding via a novel multiscale sum-product network", 2016

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

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

Analogously, we could also do a MMAP query:

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

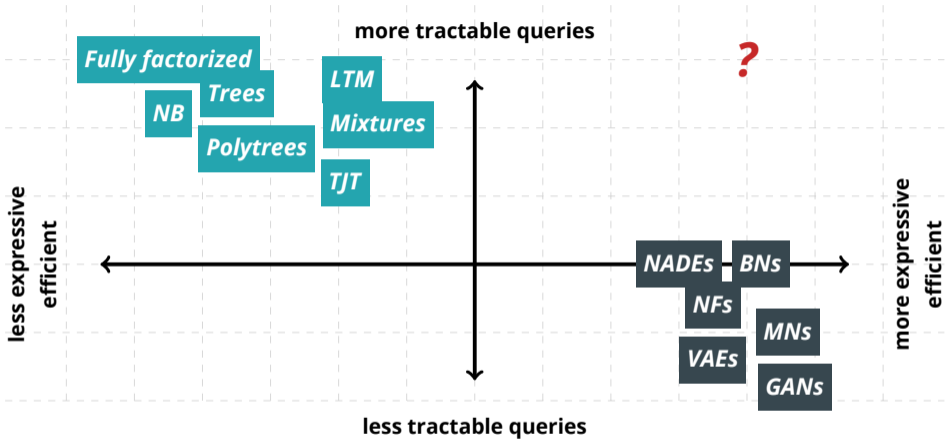
Determinism + **decomposability** = ~~tractable MMAP~~

We **cannot** decompose a MMAP query!

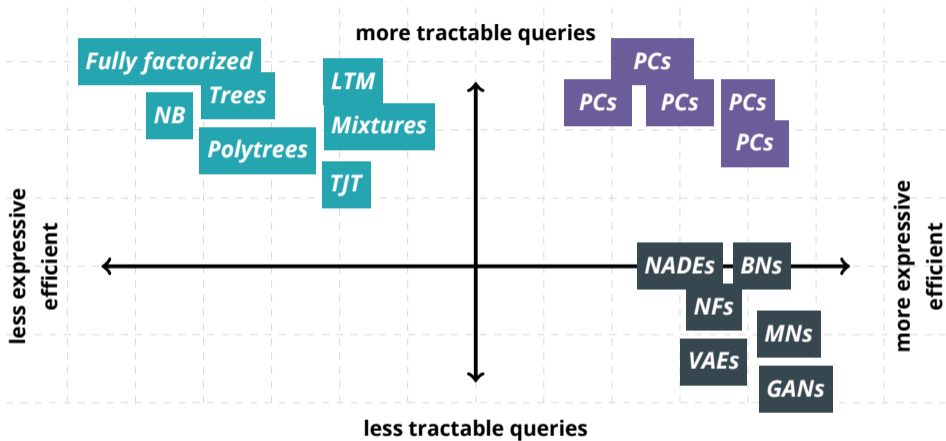
$$\operatorname{argmax}_{\mathbf{q}} \sum_{\mathbf{z}} p(\mathbf{q}, \mathbf{z} \mid \mathbf{e})$$

we still have latent variables to marginalize...

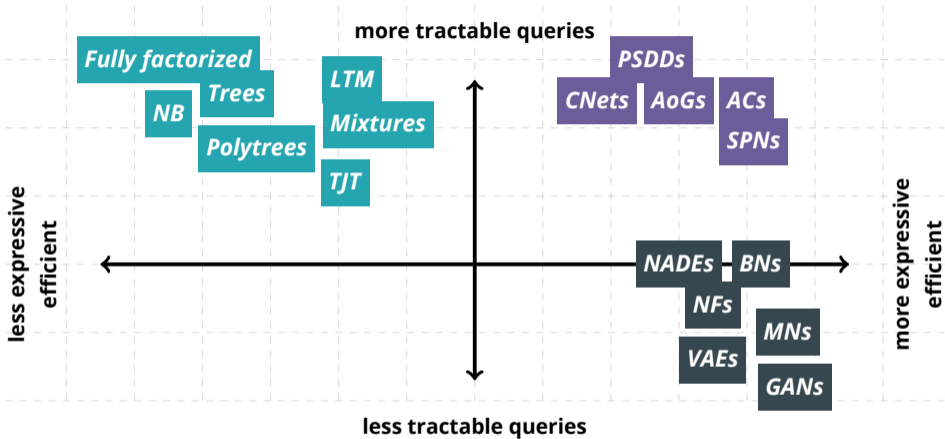
⇒ *The final part of this tutorial will talk more about advanced queries and their tractability properties.*



where are probabilistic circuits?



tractability vs expressive efficiency



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., "Random sum-product networks: A simple but effective approach to probabilistic deep learning", 2019

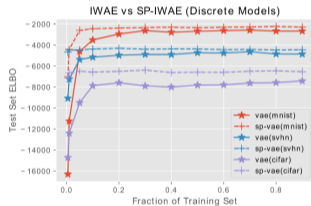
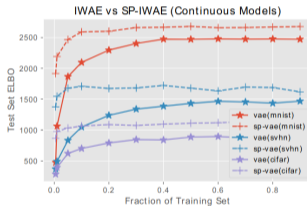
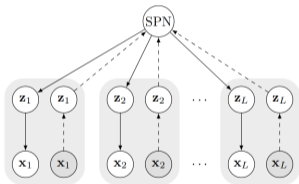
How expressive are probabilistic circuits?

density estimation benchmarks

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

Hybrid intractable + tractable EVI

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



⇒ decomposing a joint ELBO: better lower-bounds than a single VAE
⇒ more expressive efficient and less data hungry

Learning Probabilistic Circuits

Learning probabilistic circuits

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

Learning probabilistic circuits

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

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

Learning probabilistic circuits

	<i>Parameters</i>	<i>Structure</i>
<i>Generative</i>	?	?
<i>Discriminative</i>	?	?

Stay tuned for...

Next:

1. *How to learn circuit parameters?*

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

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

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

After:

How circuits are related to other tractable models?

Learning probabilistic circuits

Probabilistic circuits are (peculiar) neural networks... *just backprop with SGD!*

Learning probabilistic circuits

Probabilistic circuits are (peculiar) neural networks... ***just backprop with SGD!***

...end of Learning section!

Learning probabilistic circuits

Probabilistic circuits are (peculiar) neural networks... ***just backprop with SGD!***

wait but...

SGD is slow to converge...can we do better?

How to learn normalized weights?

Can we exploit structural properties somehow?

Learning input distributions

As simple as tossing a coin

$$\textcircled{\wedge}$$
$$X_1$$

The simplest PC: a single input distribution p_L with parameters θ

\Rightarrow *maximum likelihood (ML) estimation over data \mathcal{D}*

Learning input distributions

As simple as tossing a coin

$$\textcircled{\wedge}$$
$$X_1$$

The simplest PC: a single input distribution p_L with parameters θ

\Rightarrow maximum likelihood (ML) estimation over data \mathcal{D}

E.g. Bernoulli with parameter θ

$$\hat{\theta}_{\text{ML}} = \frac{\sum_{x \in \mathcal{D}} \mathbf{1}[x = 1] + \alpha}{|\mathcal{D}| + 2\alpha} \quad \Rightarrow \quad \text{Laplace smoothing}$$

Learning input distributions

General case: still simple

Bernoulli, Gaussian, Dirichlet, Poisson, Gamma are **exponential families** of the form:

$$p_{\mathbf{L}}(\mathbf{x}) = h(\mathbf{x}) \exp(\mathbf{T}(\mathbf{x})^T \boldsymbol{\theta} - A(\boldsymbol{\theta}))$$

Learning input distributions

General case: still simple

Bernoulli, Gaussian, Dirichlet, Poisson, Gamma are **exponential families** of the form:

$$p_{\mathbf{L}}(\mathbf{x}) = h(\mathbf{x}) \exp(\mathbf{T}(\mathbf{x})^T \boldsymbol{\theta} - A(\boldsymbol{\theta}))$$

Where:

- $A(\boldsymbol{\theta})$: log-normalizer
- $h(\mathbf{x})$ base-measure
- $\mathbf{T}(\mathbf{x})$ sufficient statistics
- $\boldsymbol{\theta}$ natural parameters

Learning input distributions

General case: still simple

Bernoulli, Gaussian, Dirichlet, Poisson, Gamma are **exponential families** of the form:

$$p_{\mathbf{L}}(\mathbf{x}) = h(\mathbf{x}) \exp(\mathbf{T}(\mathbf{x})^T \boldsymbol{\theta} - A(\boldsymbol{\theta}))$$

Where:

- $A(\boldsymbol{\theta})$: log-normalizer
- $h(\mathbf{x})$ base-measure
- $\mathbf{T}(\mathbf{x})$ sufficient statistics
- $\boldsymbol{\theta}$ natural parameters
- or $\boldsymbol{\phi}$ expectation parameters — 1:1 mapping with $\boldsymbol{\theta} \implies \boldsymbol{\theta} = \boldsymbol{\theta}(\boldsymbol{\phi})$

Learning input distributions

General case: still simple

Bernoulli, Gaussian, Dirichlet, Poisson, Gamma are **exponential families** of the form:

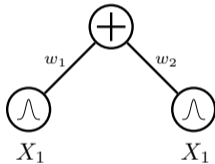
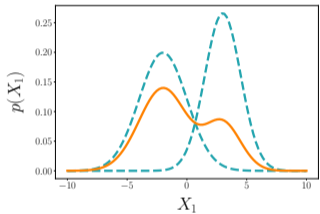
$$p_{\mathbf{L}}(\mathbf{x}) = h(\mathbf{x}) \exp(\mathbf{T}(\mathbf{x})^T \boldsymbol{\theta} - A(\boldsymbol{\theta}))$$

Maximum likelihood estimation is still “**counting**”:

$$\hat{\boldsymbol{\phi}}_{\text{ML}} = \mathbb{E}_{\mathcal{D}}[\mathbf{T}(\mathbf{x})] = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \mathbf{T}(\mathbf{x})$$

$$\hat{\boldsymbol{\theta}}_{\text{ML}} = \boldsymbol{\theta}(\hat{\boldsymbol{\phi}}_{\text{ML}})$$

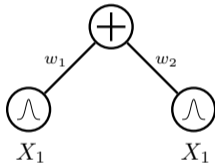
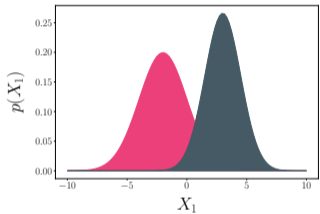
The simplest "real" PC: a sum node



Recall that sum nodes represent **mixture models**:

$$p_S(\mathbf{x}) = \sum_{k=1}^K w_k p_{L_k}(\mathbf{x})$$

The simplest "real" PC: a sum node



Recall that sum nodes represent *latent variable models*:

$$p_S(\mathbf{x}) = \sum_{k=1}^K p(Z = k)p(\mathbf{x} \mid Z = k)$$

Expectation-Maximization (EM)

Learning latent variable models: the EM recipe

Expectation-maximization = ***maximum-likelihood under missing data.***

Given: $p(\mathbf{X}, \mathbf{Z})$ where \mathbf{X} observed, \mathbf{Z} missing at random.

$$\boldsymbol{\theta}^{new} \leftarrow \arg \max_{\boldsymbol{\theta}} \mathbb{E}_{p(\mathbf{Z} | \mathbf{x}; \boldsymbol{\theta}^{old})} [\log p(\mathbf{X}, \mathbf{Z}; \boldsymbol{\theta})]$$

Expectation-Maximization for mixtures

■ $\boldsymbol{\theta}^{new} \leftarrow \arg \max_{\boldsymbol{\theta}} \mathbb{E}_{p(Z|\mathbf{x};\boldsymbol{\theta}^{old})} [\log p(\mathbf{X}, Z; \boldsymbol{\theta})]$

■ ML if Z was observed:

$$\hat{w}_k = \frac{\sum_{z \in \mathcal{D}} \mathbb{1}[z = k]}{|\mathcal{D}|} \quad \hat{\phi}_k = \frac{\sum_{\mathbf{x}, z \in \mathcal{D}} \mathbb{1}[z = k] T(\mathbf{x})}{\sum_{z \in \mathcal{D}} \mathbb{1}[z = k]}$$

■ Z is unobserved—but we have $p(Z = k | \mathbf{x}) \propto w_k L_k(\mathbf{x})$.

$$w_k^{new} = \frac{\sum_{\mathbf{x} \in \mathcal{D}} p(Z = k | \mathbf{x})}{|\mathcal{D}|} \quad \phi_k^{new} = \frac{\sum_{\mathbf{x}, z \in \mathcal{D}} p(Z = k | \mathbf{x}) T(\mathbf{x})}{\sum_{z \in \mathcal{D}} p(Z = k | \mathbf{x})}$$

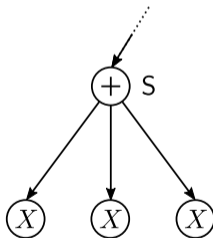
Expectation-Maximization for PCs

- EM for mixtures well understood.
- Mixtures are PCs with 1 sum node.
- The general case, PCs with many sum nodes, is similar ...

Augmentation

Making Latent Variables Explicit

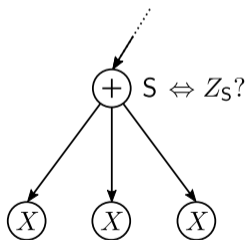
[Peharz et al. 2016]



Augmentation

Making Latent Variables Explicit

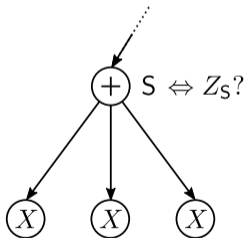
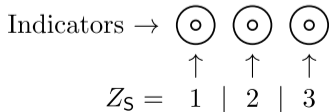
[Peharz et al. 2016]



Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

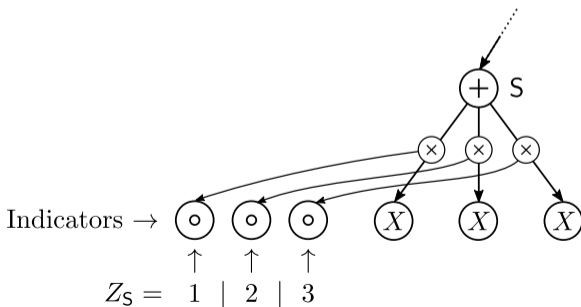


Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

Setting all indicators to 1 \Rightarrow same computation.

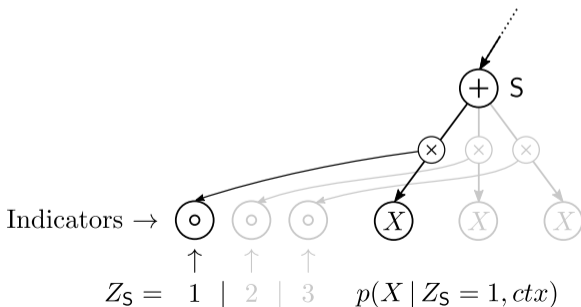


Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

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

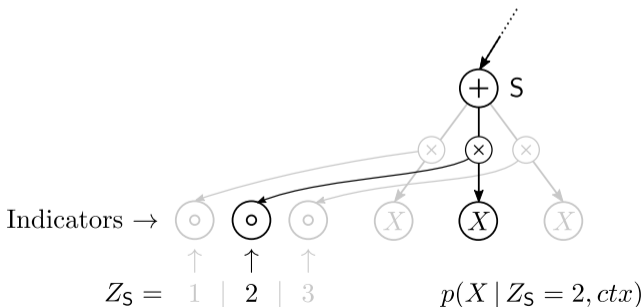


Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

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

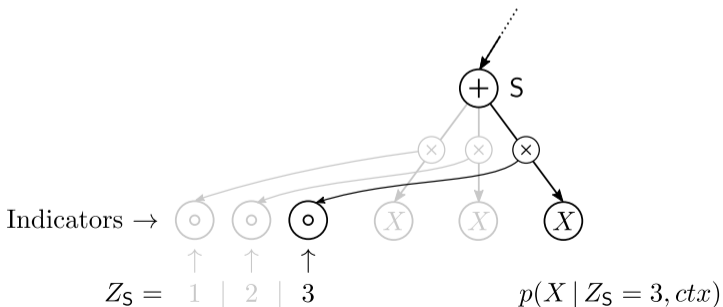


Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

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



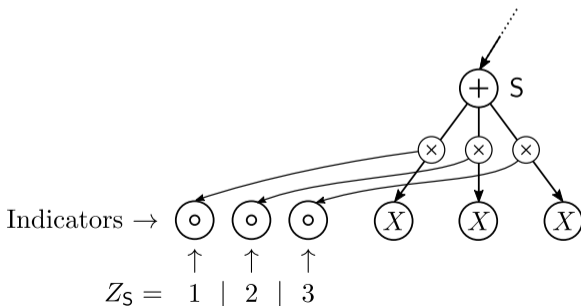
Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

Setting all indicators to 1 \Rightarrow same computation.

Have we included Z_S in the model?

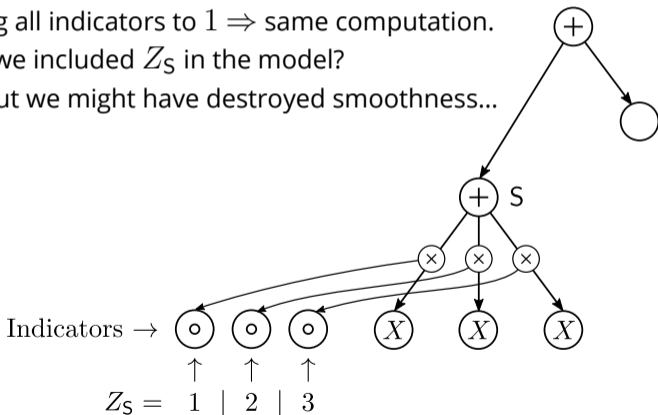


Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

Setting all indicators to 1 \Rightarrow same computation.
Have we included Z_S in the model?
Yes, but we might have destroyed smoothness...

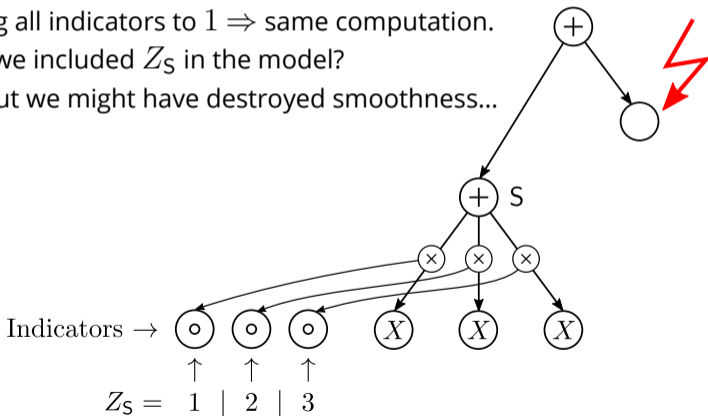


Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

Setting all indicators to 1 \Rightarrow same computation.
Have we included Z_S in the model?
Yes, but we might have destroyed smoothness...

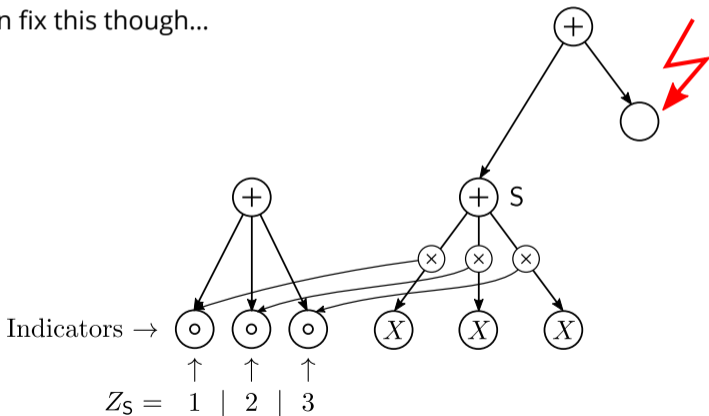


Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

We can fix this though...

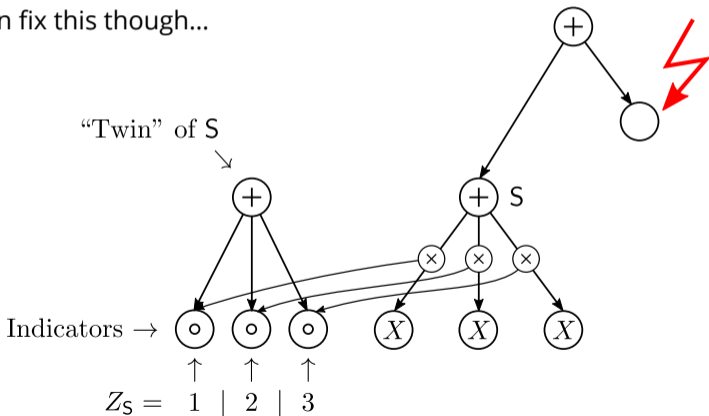


Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

We can fix this though...

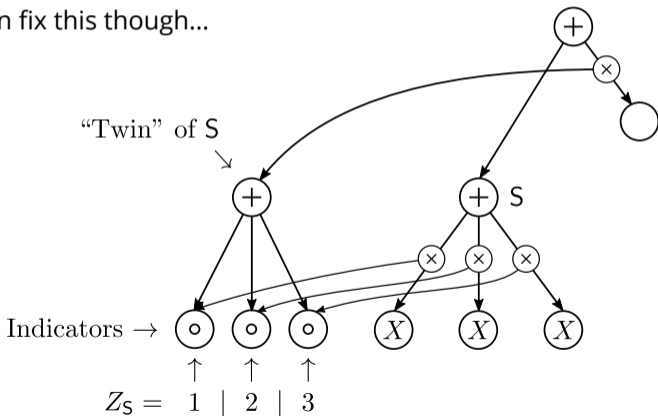


Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

We can fix this though...

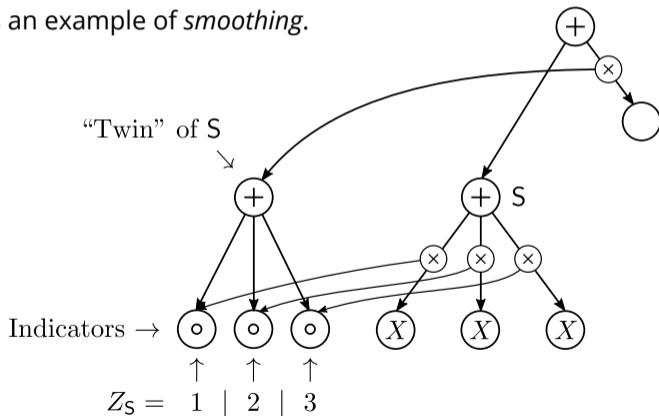


Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

This is an example of *smoothing*.

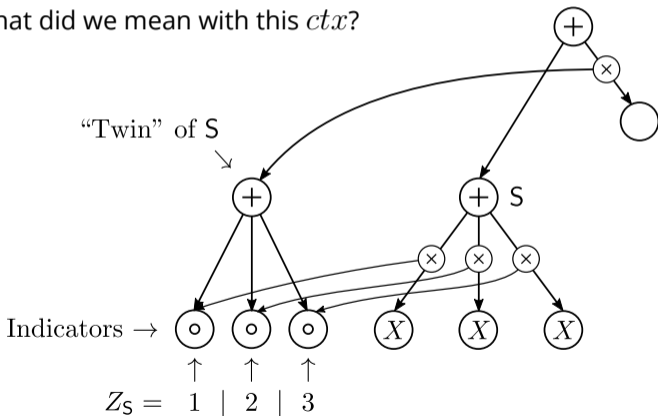


Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

But what did we mean with this *ctx*?

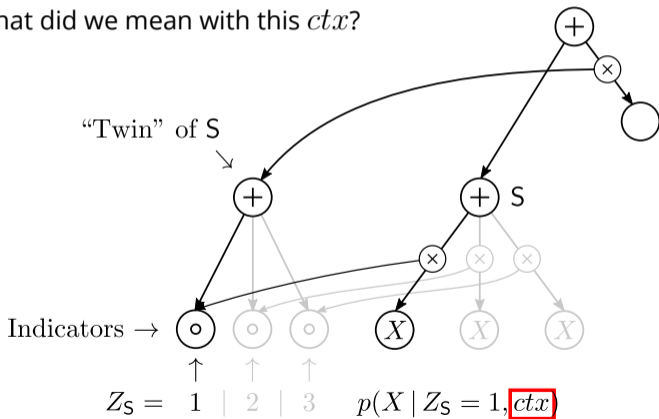


Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

But what did we mean with this *ctx*?

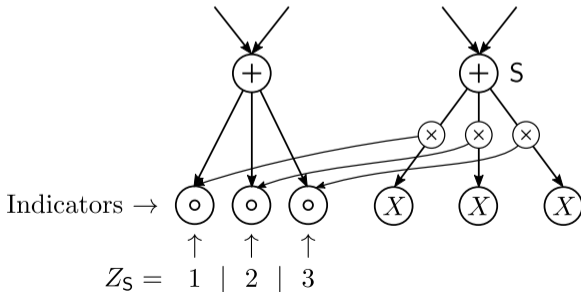


Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

One can show that the latent variables “above”...



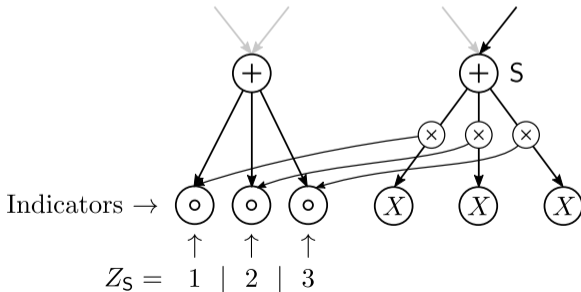
Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

...select *either* a path to S , or ...

$ctx = 1$



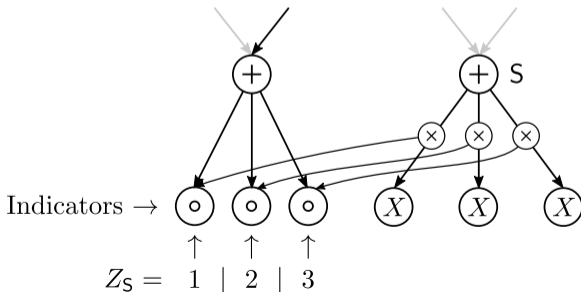
Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

...to its "twin" — but not both.

$$ctx = 0$$

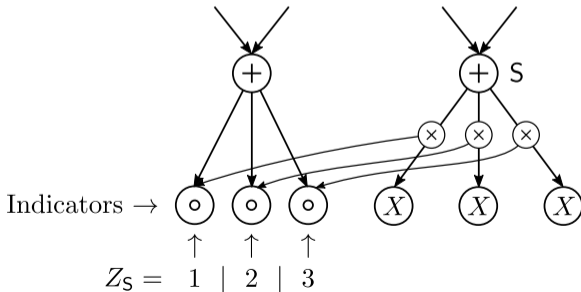


Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

Thus, sum weights have sound probabilistic semantics.



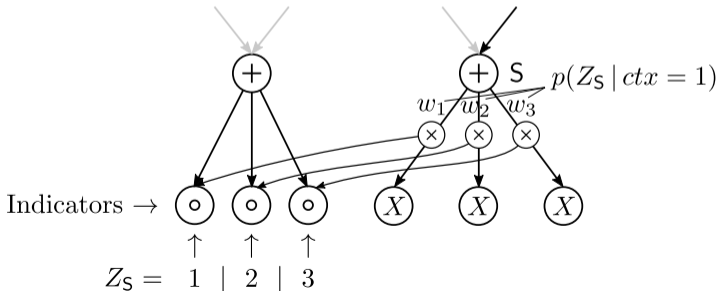
Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

Thus, sum weights have sound probabilistic semantics.

$$ctx = 1$$



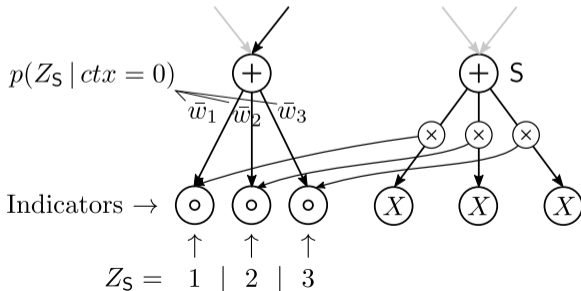
Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

Thus, sum weights have sound probabilistic semantics.

$$ctx = 0$$

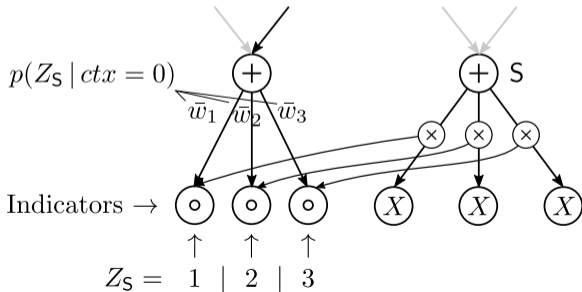


Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

Note, that when $ctx = 0$, Z_S becomes **independent** of X !

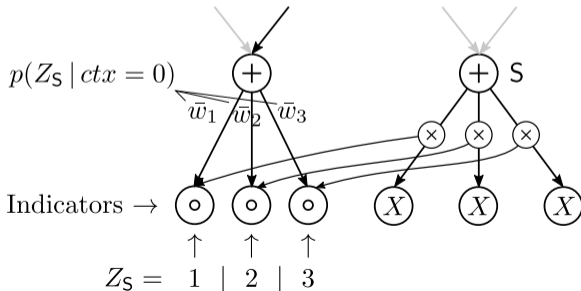


Augmentation

Making Latent Variables Explicit

[Peharz et al. 2016]

Note, that when $ctx = 0$, Z_S becomes **independent** of X !
Thus, $\bar{w}_1, \bar{w}_2, \bar{w}_3$ can be set arbitrary.



Augmentation

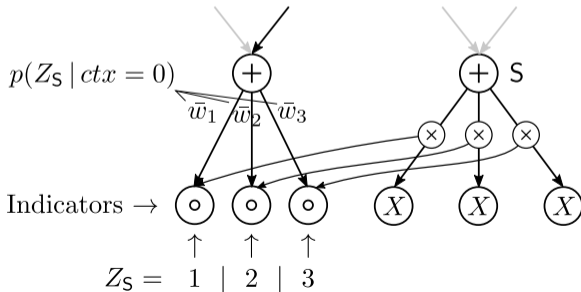
Making Latent Variables Explicit

[Peharz et al. 2016]

Note, that when $ctx = 0$, Z_S becomes **independent** of X !

Thus, $\bar{w}_1, \bar{w}_2, \bar{w}_3$ can be set arbitrary.

Do we need to store them then?



Augmentation

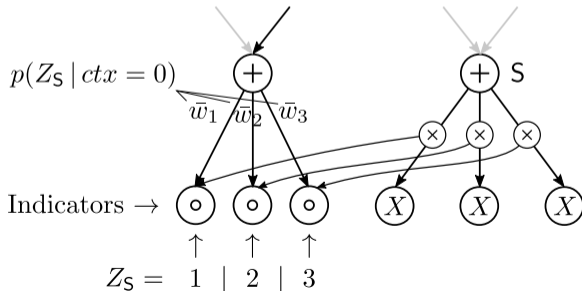
Making Latent Variables Explicit

[Peharz et al. 2016]

Note, that when $ctx = 0$, Z_S becomes **independent** of X !

Thus, $\bar{w}_1, \bar{w}_2, \bar{w}_3$ can be set arbitrary.

Do we need to store them then? **No!**

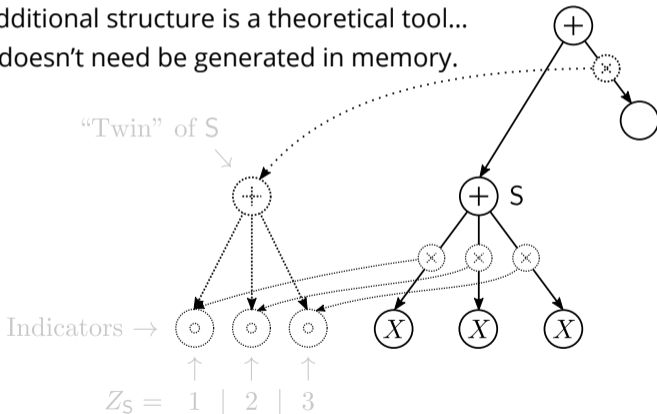


Augmentation

Making Latent Variables Explicit

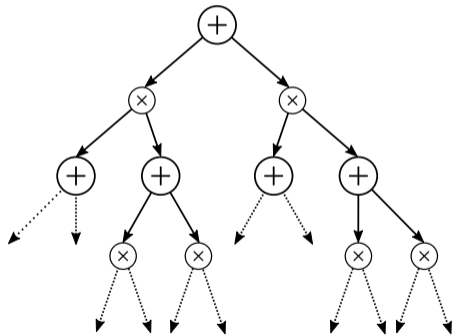
[Peharz et al. 2016]

This additional structure is a theoretical tool...
...and doesn't need be generated in memory.



Expectation-Maximization

Tractable MAR (smooth, decomposable)

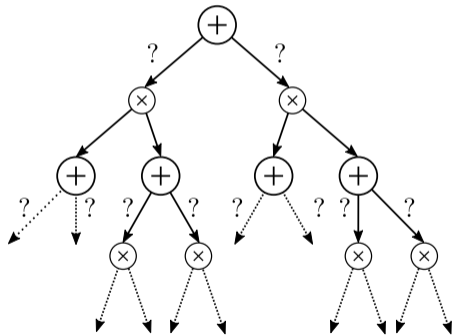


For learning, we need to know
for each sum S :

1. Is S reached ($ctx = ?$)
2. Which child does it select ($Z_S = ?$)

Expectation-Maximization

Tractable MAR (smooth, decomposable)

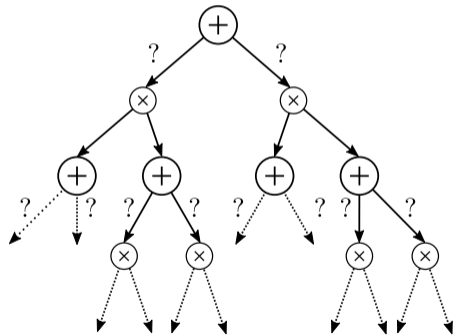


For learning, we need to know for each sum S :

1. Is S reached ($ctx = ?$)
2. Which child does it select ($Z_S = ?$)

Expectation-Maximization

Tractable MAR (smooth, decomposable)



For learning, we need to know for each sum S :

1. Is S reached ($ctx = ?$)
2. Which child does it select ($Z_S = ?$)

We can **infer** it: $p(ctx, Z_S | \mathbf{x})$

Expectation-Maximization

Tractable MAR (smooth, decomposable)

$$w_{i,j}^{new} \leftarrow \frac{\sum_{\mathbf{x} \in \mathcal{D}} p[ctx_i = 1, Z_i = j \mid \mathbf{x}; \mathbf{w}^{old}]}{\sum_{\mathbf{x} \in \mathcal{D}} p[ctx_i = 1 \mid \mathbf{x}; \mathbf{w}^{old}]}$$

Darwiche, "A Differential Approach to Inference in Bayesian Networks", 2003

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

Expectation-Maximization

Tractable MAR (smooth, decomposable)

$$w_{i,j}^{new} \leftarrow \frac{\sum_{\mathbf{x} \in \mathcal{D}} p[ctx_i = 1, Z_i = j \mid \mathbf{x}; \mathbf{w}^{old}]}{\sum_{\mathbf{x} \in \mathcal{D}} p[ctx_i = 1 \mid \mathbf{x}; \mathbf{w}^{old}]}$$

We get **all** the required statistics with a single backprop pass:

$$p[ctx_i = 1, Z_i = j \mid \mathbf{x}; \mathbf{w}^{old}] = \frac{1}{p(\mathbf{x})} \frac{\partial p(\mathbf{x})}{\partial S_i(\mathbf{x})} \mathbf{N}_j(\mathbf{x}) w_{i,j}^{old}$$

Darwiche, "A Differential Approach to Inference in Bayesian Networks", 2003

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

Expectation-Maximization

Tractable MAR (smooth, decomposable)

$$w_{i,j}^{new} \leftarrow \frac{\sum_{\mathbf{x} \in \mathcal{D}} p[ctx_i = 1, Z_i = j \mid \mathbf{x}; \mathbf{w}^{old}]}{\sum_{\mathbf{x} \in \mathcal{D}} p[ctx_i = 1 \mid \mathbf{x}; \mathbf{w}^{old}]}$$

We get **all** the required statistics with a single backprop pass:

$$p[ctx_i = 1, Z_i = j \mid \mathbf{x}; \mathbf{w}^{old}] = \frac{1}{p(\mathbf{x})} \frac{\partial p(\mathbf{x})}{\partial S_i(\mathbf{x})} \mathbf{N}_j(\mathbf{x}) w_{i,j}^{old}$$

\Rightarrow This also works with missing values in \mathbf{x} !

Darwiche, "A Differential Approach to Inference in Bayesian Networks", 2003

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

Expectation-Maximization

Tractable MAR (smooth, decomposable)

$$w_{i,j}^{new} \leftarrow \frac{\sum_{\mathbf{x} \in \mathcal{D}} p[ctx_i = 1, Z_i = j \mid \mathbf{x}; \mathbf{w}^{old}]}{\sum_{\mathbf{x} \in \mathcal{D}} p[ctx_i = 1 \mid \mathbf{x}; \mathbf{w}^{old}]}$$

We get **all** the required statistics with a single backprop pass:

$$p[ctx_i = 1, Z_i = j \mid \mathbf{x}; \mathbf{w}^{old}] = \frac{1}{p(\mathbf{x})} \frac{\partial p(\mathbf{x})}{\partial S_i(\mathbf{x})} \mathbf{N}_j(\mathbf{x}) w_{i,j}^{old}$$

\Rightarrow Similar updates for leaves, when in exponential family.

Darwiche, "A Differential Approach to Inference in Bayesian Networks", 2003

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

Expectation-Maximization

Tractable MAR (smooth, decomposable)

$$w_{i,j}^{new} \leftarrow \frac{\sum_{\mathbf{x} \in \mathcal{D}} p[ctx_i = 1, Z_i = j \mid \mathbf{x}; \mathbf{w}^{old}]}{\sum_{\mathbf{x} \in \mathcal{D}} p[ctx_i = 1 \mid \mathbf{x}; \mathbf{w}^{old}]}$$

We get **all** the required statistics with a single backprop pass:

$$p[ctx_i = 1, Z_i = j \mid \mathbf{x}; \mathbf{w}^{old}] = \frac{1}{p(\mathbf{x})} \frac{\partial p(\mathbf{x})}{\partial S_i(\mathbf{x})} \mathbf{N}_j(\mathbf{x}) w_{i,j}^{old}$$

\Rightarrow also derivable from a concave-convex procedure (CCCP) [Zhao et al. 2016a]

Darwiche, "A Differential Approach to Inference in Bayesian Networks", 2003

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

Expectation-Maximization

Tractable MAR/MAP (smooth, decomposable, deterministic)

~~Expectation Maximization~~ Exact ML

Tractable MAR/MAP (smooth, decomposable, deterministic)

Exact ML

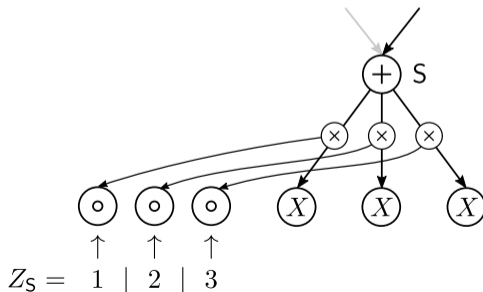
Tractable MAR/MAP (smooth, decomposable, deterministic)

Deterministic circuit \Rightarrow at most one non-zero sum child (for complete input).

Exact ML

Tractable MAR/MAP (smooth, decomposable, deterministic)

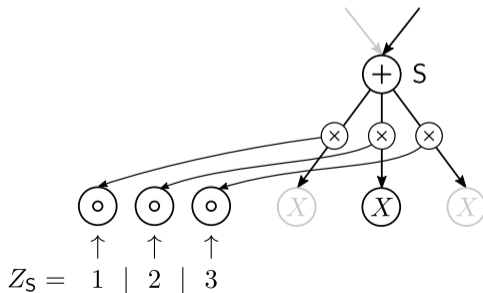
Deterministic circuit \Rightarrow at most one non-zero sum child (for complete input).



Exact ML

Tractable MAR/MAP (smooth, decomposable, deterministic)

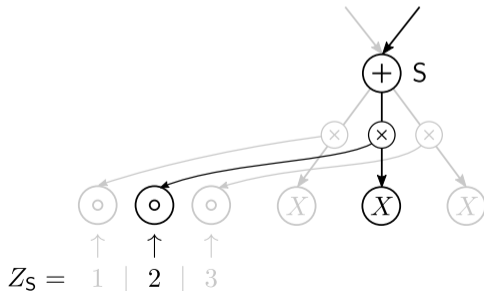
E.g., the second child of this sum node...



Exact ML

Tractable MAR/MAP (smooth, decomposable, deterministic)

...but that rules out $Z_S \in \{1, 3\}$! $\Rightarrow Z_S = 2$

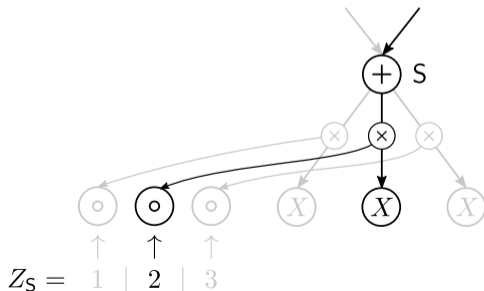


Exact ML

Tractable MAR/MAP (smooth, decomposable, deterministic)

...but that rules out $Z_S \in \{1, 3\}$! $\Rightarrow Z_S = 2$

Thus, the latent variables are **actually observed** in deterministic circuits!



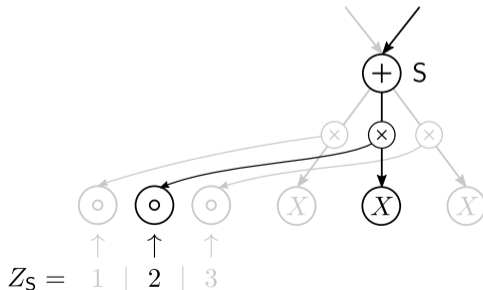
Exact ML

Tractable MAR/MAP (smooth, decomposable, deterministic)

...but that rules out $Z_S \in \{1, 3\}$! $\Rightarrow Z_S = 2$

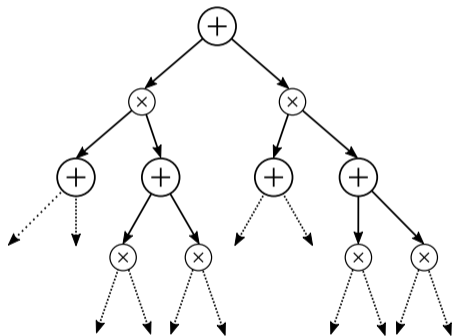
Thus, the latent variables are **actually observed** in deterministic circuits!

They are (deterministic) functions of the observed data.



Example

Tractable MAR/MAP (smooth, decomposable, deterministic)

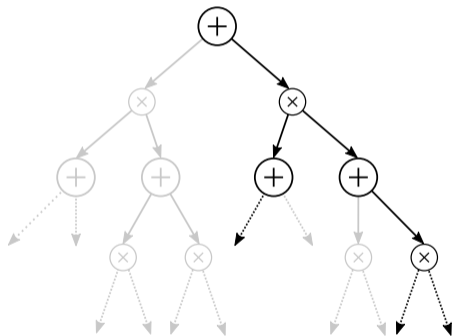


For each sum node, we know

1. if it is reached ($ctx = 1$)
2. which child it selects

Example

Tractable MAR/MAP (smooth, decomposable, deterministic)

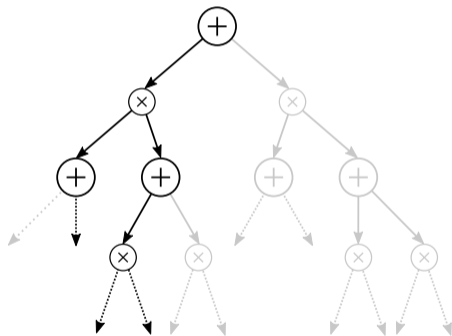


For each sum node, we know

1. if it is reached ($ctx = 1$)
2. which child it selects

Example

Tractable MAR/MAP (smooth, decomposable, deterministic)

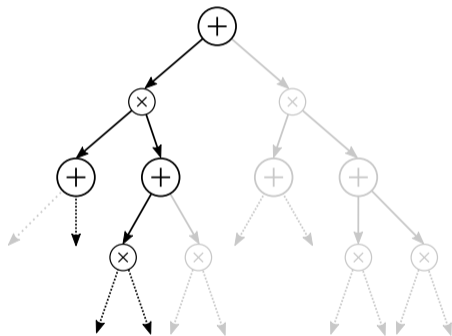


For each sum node, we know

1. if it is reached ($ctx = 1$)
2. which child it selects

Example

Tractable MAR/MAP (smooth, decomposable, deterministic)



For each sum node, we know

1. if it is reached ($ctx = 1$)
2. which child it selects

⇒ **MLE by counting!**

Exact ML

Tractable MAR/MAP (smooth, decomposable, deterministic)

Given a complete dataset \mathcal{D} , the maximum-likelihood sum-weights are:

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

Kisa et al., "Probabilistic sentential decision diagrams", 2014

Pecharz et al., "Learning Selective Sum-Product Networks", 2014

Exact ML

Tractable MAR/MAP (smooth, decomposable, deterministic)

Given a complete dataset \mathcal{D} , the maximum-likelihood sum-weights are:

$$w_{i,j}^{\text{ML}} = \frac{\sum_{\mathbf{x} \in \mathcal{D}} \mathbb{1}\{\mathbf{x} \models [i \wedge j]\}}{\sum_{\mathbf{x} \in \mathcal{D}} \mathbb{1}\{\mathbf{x} \models [i]\}} \quad \leftarrow \text{ctx}_i = 1, Z_i = j$$

Kisa et al., "Probabilistic sentential decision diagrams", 2014

Pecharz et al., "Learning Selective Sum-Product Networks", 2014

Exact ML

Tractable MAR/MAP (smooth, decomposable, deterministic)

Given a complete dataset \mathcal{D} , the maximum-likelihood sum-weights are:

$$w_{i,j}^{\text{ML}} = \frac{\sum_{\mathbf{x} \in \mathcal{D}} \mathbb{1}\{\mathbf{x} \models [i \wedge j]\}}{\sum_{\mathbf{x} \in \mathcal{D}} \mathbb{1}\{\mathbf{x} \models [i]\}} \quad \begin{array}{l} \leftarrow \text{ctx}_i = 1, Z_i = j \\ \leftarrow \text{ctx}_i = 1 \end{array}$$

Kisa et al., "Probabilistic sentential decision diagrams", 2014

Pecharz et al., "Learning Selective Sum-Product Networks", 2014

Exact ML

Tractable MAR/MAP (smooth, decomposable, deterministic)

Given a complete dataset \mathcal{D} , the maximum-likelihood sum-weights are:

$$w_{i,j}^{\text{ML}} = \frac{\sum_{\mathbf{x} \in \mathcal{D}} \mathbb{1}\{\mathbf{x} \models [i \wedge j]\}}{\sum_{\mathbf{x} \in \mathcal{D}} \mathbb{1}\{\mathbf{x} \models [i]\}} \quad \begin{array}{l} \leftarrow \text{ctx}_i = 1, Z_i = j \\ \leftarrow \text{ctx}_i = 1 \end{array}$$

- \Rightarrow regularization, e.g. Laplace-smoothing, to avoid division by zero
- \Rightarrow global maximum with single pass over \mathcal{D}
- \Rightarrow when missing data, fallback to EM

Kisa et al., "Probabilistic sentential decision diagrams", 2014

Pecharz et al., "Learning Selective Sum-Product Networks", 2014

Bayesian parameter learning

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

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

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

Learning probabilistic circuits

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

Image-tailored (handcrafted) structures

“Recursive Image Slicing”



Image-tailored (handcrafted) structures

"Recursive Image Slicing"

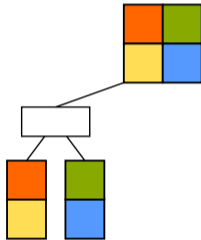


Image-tailored (handcrafted) structures

"Recursive Image Slicing"

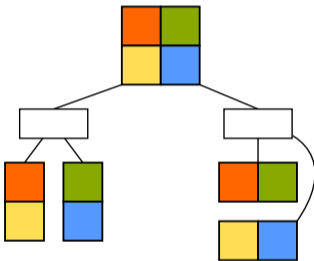


Image-tailored (handcrafted) structures

"Recursive Image Slicing"

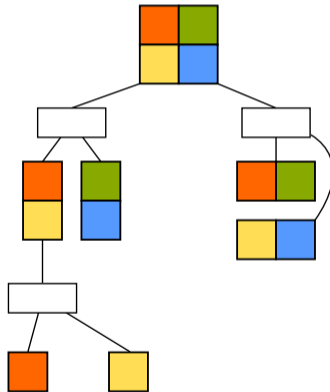


Image-tailored (handcrafted) structures

"Recursive Image Slicing"

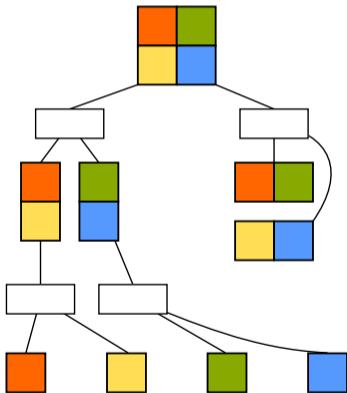


Image-tailored (handcrafted) structures

"Recursive Image Slicing"

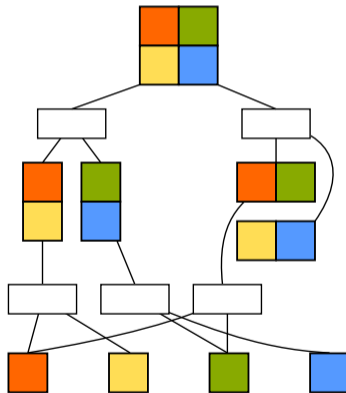


Image-tailored (handcrafted) structures

"Recursive Image Slicing"

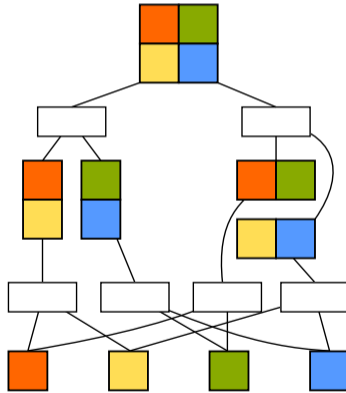


Image-tailored (handcrafted) structures

"Recursive Image Slicing"

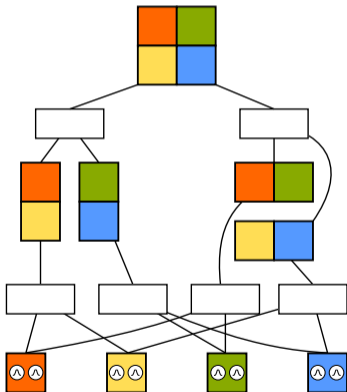


Image-tailored (handcrafted) structures

"Recursive Image Slicing"

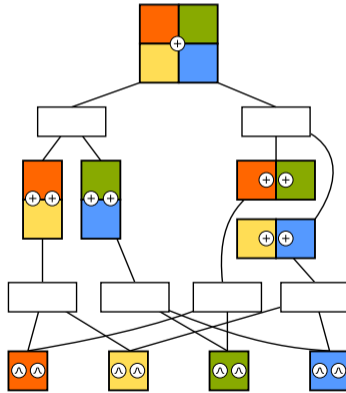


Image-tailored (handcrafted) structures

"Recursive Image Slicing"

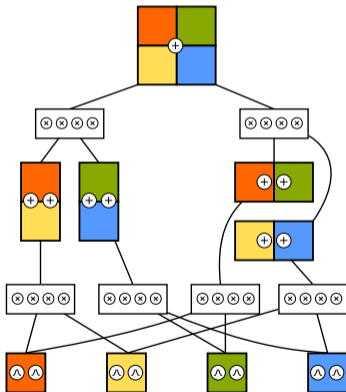


Image-tailored (handcrafted) structures

"Recursive Image Slicing"

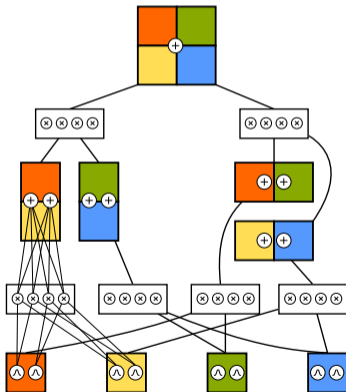


Image-tailored (handcrafted) structures

"Recursive Image Slicing"

⇒ Smooth & Decomposable

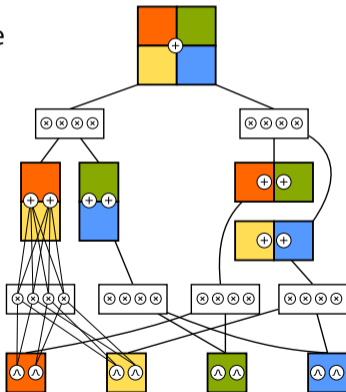
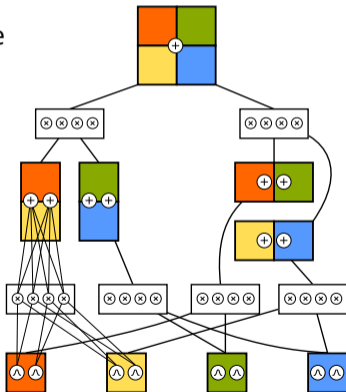


Image-tailored (handcrafted) structures

"Recursive Image Slicing"

⇒ Smooth & Decomposable

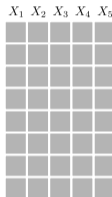
⇒ Tractable MAR



Learning the structure from data

“Recursive Data Slicing” — LearnSPN

Cluster



Learning the structure from data

“Recursive Data Slicing” — LearnSPN

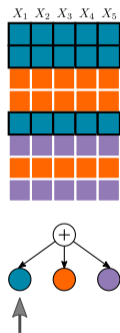
Cluster → **sum node**



Learning the structure from data

“Recursive Data Slicing” — LearnSPN

Try to find independent groups
of random variables

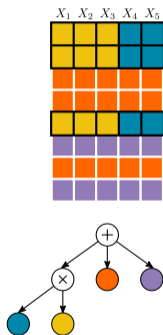


Learning the structure from data

“Recursive Data Slicing” — LearnSPN

Try to find independent groups
of random variables

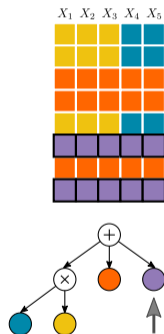
Success → **product node**



Learning the structure from data

“Recursive Data Slicing” — LearnSPN

Try to find independent groups
of random variables

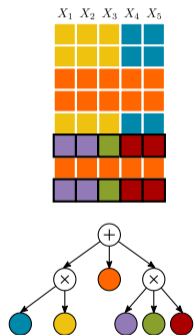


Learning the structure from data

“Recursive Data Slicing” — LearnSPN

Try to find independent groups
of random variables

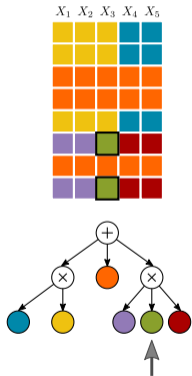
Success → **product node**



Learning the structure from data

"Recursive Data Slicing" — LearnSPN

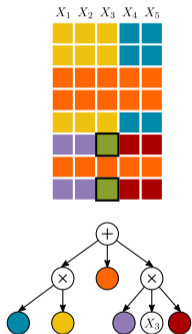
Single variable



Learning the structure from data

“Recursive Data Slicing” — LearnSPN

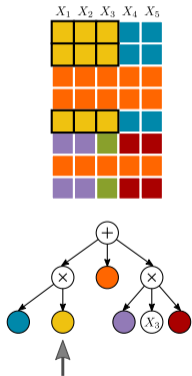
Single variable \rightarrow **leaf**



Learning the structure from data

“Recursive Data Slicing” — LearnSPN

Try to find independent groups
of random variables

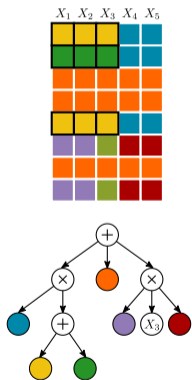


Learning the structure from data

“Recursive Data Slicing” — LearnSPN

Try to find independent groups
of random variables

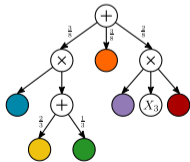
Fail → cluster → **sum node**



Learning the structure from data

“Recursive Data Slicing” — LearnSPN

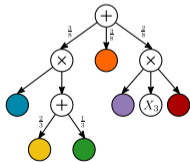
- ⇒ Continue until no further leaf can be expanded.
- ⇒ Clustering ratios also deliver (initial) parameters.



Learning the structure from data

“Recursive Data Slicing” — LearnSPN

- ⇒ Continue until no further leaf can be expanded.
- ⇒ Clustering ratios also deliver (initial) parameters.
- ⇒ Smooth & Decomposable
- ⇒ Tractable MAR



LearnSPN

Variants

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

Structure Learning + MAP (determinism)

“Recursive conditioning” — Cutset Networks

[Rahman et al. 2014]

A B C D E F

Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks

[Rahman et al. 2014]

A B C D E F



Select Variable

Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks

[Rahman et al. 2014]

A B C D E F

A

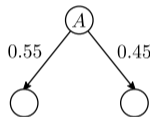
Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks

[Rahman et al. 2014]

A B C D E F

Split states



Structure Learning + MAP (determinism)

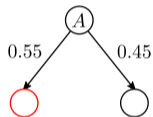
"Recursive conditioning" — Cutset Networks

[Rahman et al. 2014]

A B C D E F



Select Variable



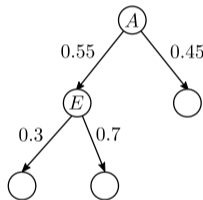
Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks

[Rahman et al. 2014]

$A B C D E F$

Split states



Structure Learning + MAP (determinism)

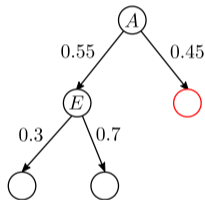
"Recursive conditioning" — Cutset Networks

[Rahman et al. 2014]

A B C D E F



Select Variable



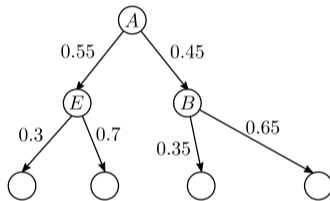
Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks

[Rahman et al. 2014]

A B C D E F

Split states



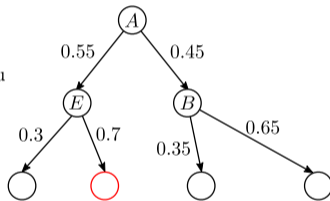
Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks

[Rahman et al. 2014]

$A B C D E F$

Stop → learn Chow-Liu

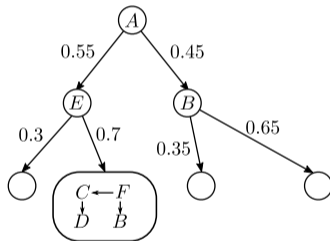


Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks

[Rahman et al. 2014]

$A B C D E F$



Structure Learning + MAP (determinism)

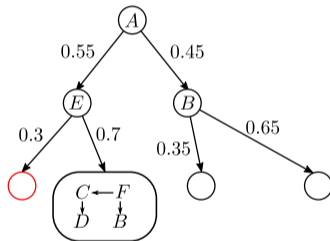
"Recursive conditioning" — Cutset Networks

[Rahman et al. 2014]

A B C D E F



Select Variable



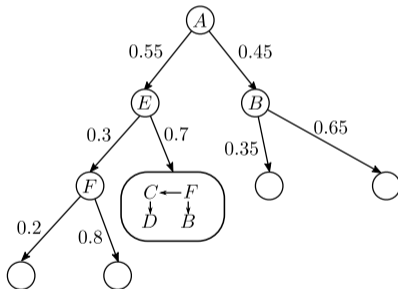
Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks

[Rahman et al. 2014]

$A B C D E F$

Split states



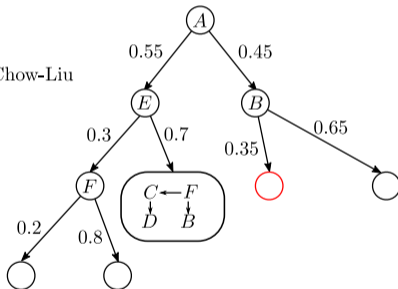
Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks

[Rahman et al. 2014]

A B C D E F

Stop → learn Chow-Liu

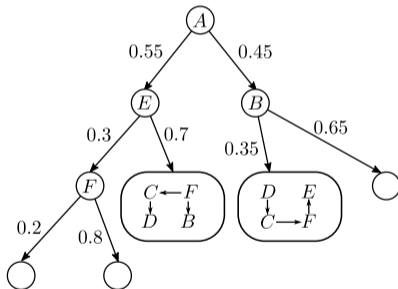


Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks

[Rahman et al. 2014]

A B C D E F

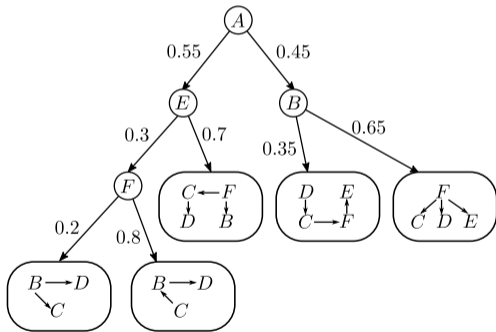


Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks

[Rahman et al. 2014]

...and so on.

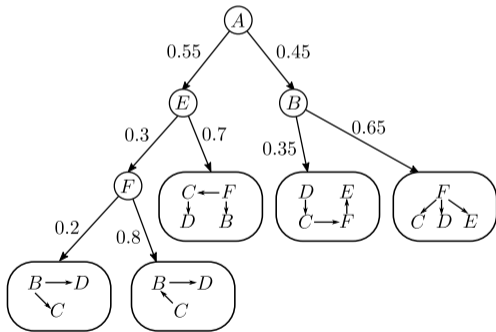


Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks

[Rahman et al. 2014]

Convert into PC...

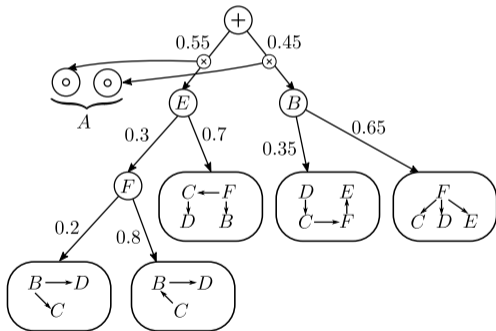


Structure Learning + MAP (determinism)

“Recursive conditioning” — Cutset Networks

[Rahman et al. 2014]

Convert into PC...

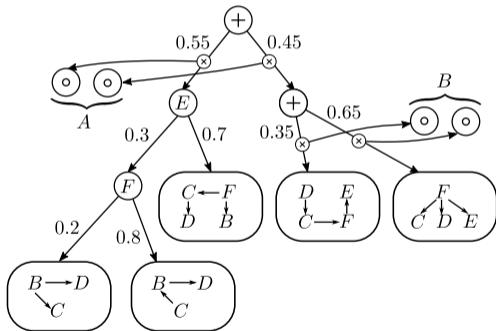


Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks

[Rahman et al. 2014]

Convert into PC...

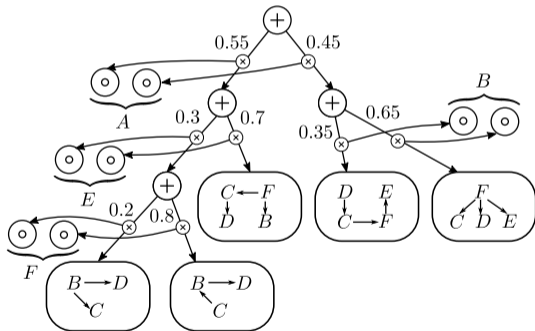


Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks

[Rahman et al. 2014]

Convert into PC...

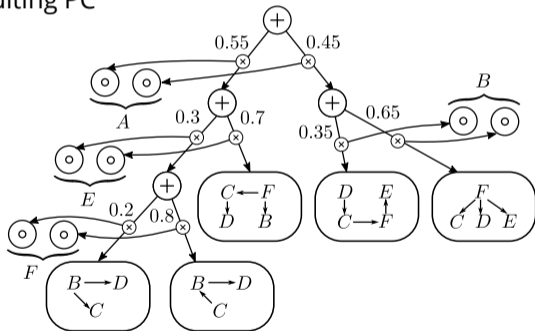


Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks

[Rahman et al. 2014]

Convert into PC... Resulting PC
is deterministic.



Cutset networks (CNets)

Variants

- Variable selection based on entropy [*Rahman et al. 2014*]
- Can be extended to mixtures of CNets using EM [*ibid.*]
- Structure search over OR-graphs/CL-trees [*Di Mauro et al. 2015a*]
- Boosted CNets [*Rahman et al. 2016*]
- Randomized CNets, Bagging [*Di Mauro et al. 2017*]

Structure learning + MAP (determinism)

Greedy structure search

[Peharz2014; Lowd et al. 2008; Liang et al. 2017a]

- Structure learning as discrete optimization
- Typical objective:

$$\mathcal{O} = \log \mathcal{L} + \lambda |\mathcal{C}|,$$

where $\log \mathcal{L}$ is log-likelihood using ML-parameters, and $|\mathcal{C}|$ the PC's size (\Leftrightarrow worst case inference cost).

- Iterate:
 1. Start with a simple initial structure.
 2. Perform local structure modifications, greedily improving \mathcal{O}

Randomized structure learning

Extremely Randomized C Nets (XC Nets) *[Di Mauro et al. 2017]*

- Top-down random conditioning.
- Learning Chow-Liu trees at the leaves.
- Smooth, decomposable, deterministic.

Random Tensorized SPNs (RAT-SPNs) *[Peharz et al. 2019a]*

- Random tree-shaped PCs.
- Discriminative+generative parameter learning (SGD/EM + dropout).
- Smooth, decomposable.

Ensembles of probabilistic circuits

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

Solution: *ensembles of circuits!*

⇒ *non-deterministic mixture models: another sum node!*

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

Ensemble weights and components can be learned separately or jointly

- EM or structural EM
- bagging
- boosting

Bagging

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

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

- more efficient than bagging

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

Di Mauro et al., "Learning Bayesian Random Cutset Forests", 2015

Boosting

Boosting Probabilistic Circuits

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

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

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

Learning probabilistic circuits

Parameters

Structure

Generative

deterministic

closed-form MLE [Kisa et al. 2014a; Peharz et al. 2014]

non-deterministic

EM [Poon et al. 2011; Peharz 2015; Zhao et al. 2016a]

SGD [Sharir et al. 2016; Peharz et al. 2019a]

Bayesian [Jaini et al. 2016; Rashwan et al. 2016]

[Zhao et al. 2016b; Trapp et al. 2019; Vergari et al. 2019]

greedy

top-down [Gens et al. 2013; Rooshenas et al. 2014]

[Rahman et al. 2014; Vergari et al. 2015]

bottom-up [Peharz et al. 2013]

hill climbing [Lowd et al. 2008, 2013; Peharz et al. 2014]

[Dennis et al. 2015; Liang et al. 2017a]

random RAT-SPNs [Peharz et al. 2019a] XCNet [Di Mauro et al. 2017]

Discriminative

?

?

EVI inference : ***density estimation***

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

Learning probabilistic circuits

Parameters

Structure

Generative

deterministic

closed-form MLE [Kisa et al. 2014a; Peharz et al. 2014]

non-deterministic

EM [Poon et al. 2011; Peharz 2015; Zhao et al. 2016a]

SGD [Sharir et al. 2016; Peharz et al. 2019a]

Bayesian [Jaini et al. 2016; Rashwan et al. 2016]

[Zhao et al. 2016b; Trapp et al. 2019; Vergari et al. 2019]

greedy

top-down [Gens et al. 2013; Rooshenas et al. 2014]

[Rahman et al. 2014; Vergari et al. 2015]

bottom-up [Peharz et al. 2013]

hill climbing [Lowd et al. 2008, 2013; Peharz et al. 2014]

[Dennis et al. 2015; Liang et al. 2017a]

random RAT-SPNs [Peharz et al. 2019a] XCNet [Di Mauro et al. 2017]

Discriminative

deterministic

convex-opt MLE [Liang et al. 2019]

non-deterministic

EM [Rashwan et al. 2018]

SGD [Gens et al. 2012; Sharir et al. 2016]

[Peharz et al. 2019a]

greedy

top-down [Shao et al. 2019]

hill climbing [Rooshenas et al. 2016]

Representations and theory

Stay tuned for...

Next:

1. *How are probabilistic circuits related to logical ones?*
⇒ *a historical perspective*
2. *How classical tractable models can be turned in a circuit?*
⇒ *Compiling low-treewidth PGMs*
3. *How do PCs in the literature relate and differ?*
⇒ *SPNs, ACs, CNetS, PSDDs*

After:

More advanced query classes and structural properties!

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:

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

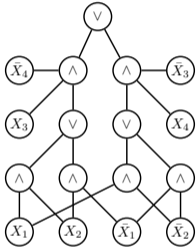
\Rightarrow Algebraic model counting [Kimmig et al. 2017], Semi-ring programming [Belle et al. 2016]

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

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

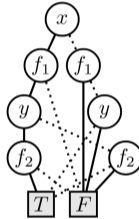
\Rightarrow logical circuits!

Logical circuits



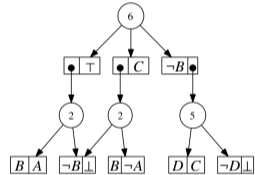
s/d-D/NNFs

[Darwiche et al. 2002a]



O/BDDs

[Bryant 1986]



SDDs

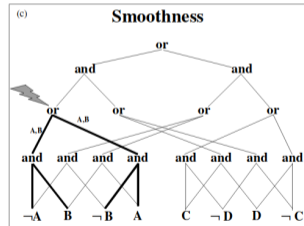
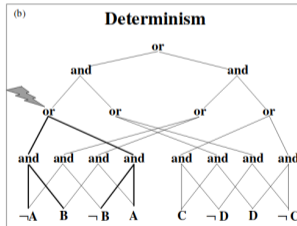
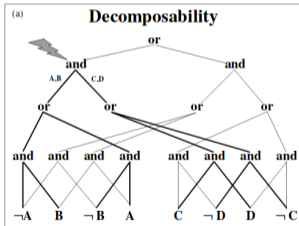
[Darwiche 2011]

Logical circuits are compact representations for boolean functions...

Logical circuits

structural properties

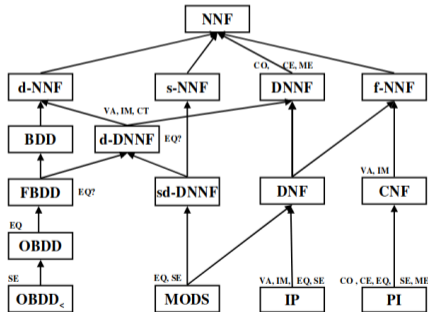
...and like probabilistic circuits, one can define **structural properties**: (structured) decomposability, smoothness, determinism allowing for tractable computations



Logical circuits

a knowledge compilation map

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



Logical circuits

connection to probabilistic circuits through WMC

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

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

■ Probabilistic inference by WMC:

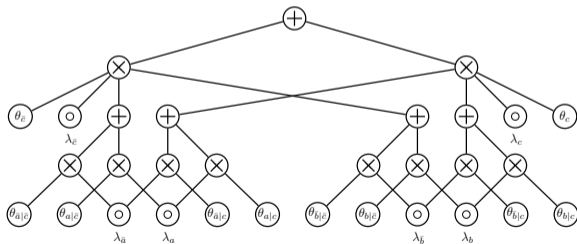
1. Encode probabilistic model as WMC formula Δ
2. Compile Δ into a logical circuit (e.g. d-DNNF, OBDD, SDD, etc.)
3. Tractable MAR/CON by tractable WMC on circuit
4. Answer complex queries tractably by enforcing more structural properties

Logical circuits

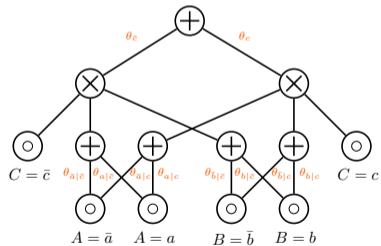
connection to probabilistic circuits through WMC

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

\Rightarrow parameter variables \rightarrow edge parameters



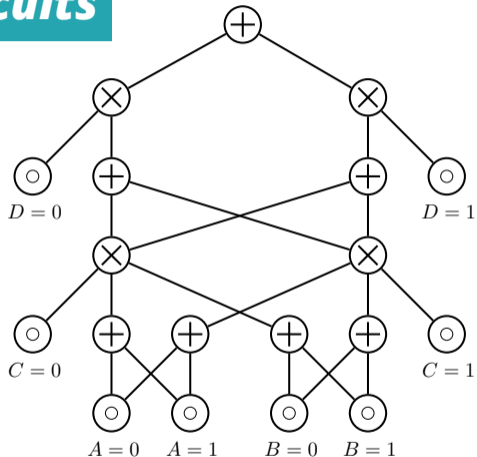
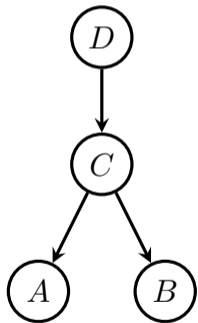
Compiled circuit of WMC encoding



Equivalent probabilistic circuit

From BN trees to circuits

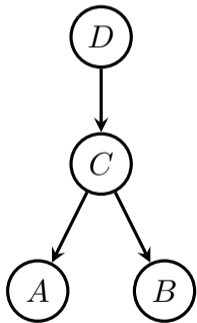
via compilation



From BN trees to circuits

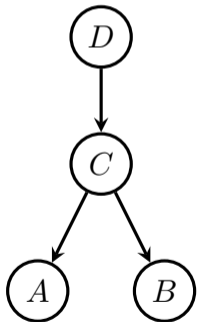
via compilation

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



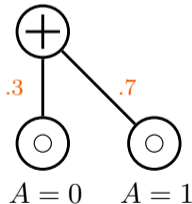
From BN trees to circuits

via compilation



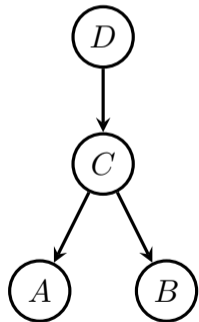
...compile a leaf CPT

$p(A|C = 0)$

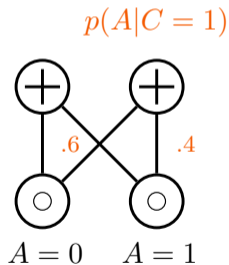


From BN trees to circuits

via compilation



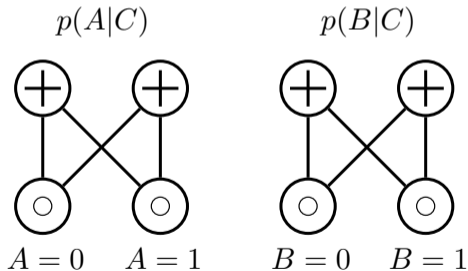
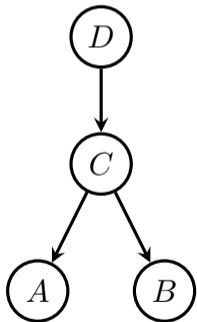
...compile a leaf CPT



From BN trees to circuits

via compilation

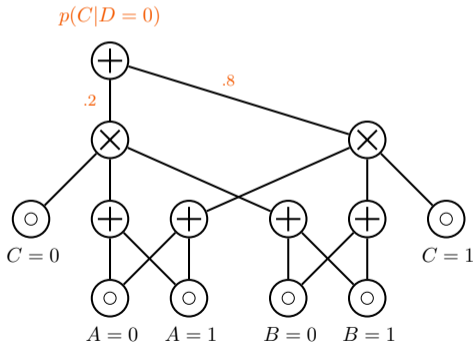
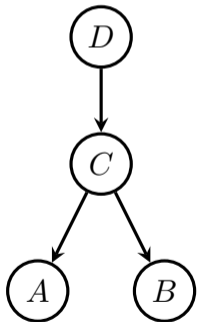
...compile a leaf CPT...for all leaves...



From BN trees to circuits

via compilation

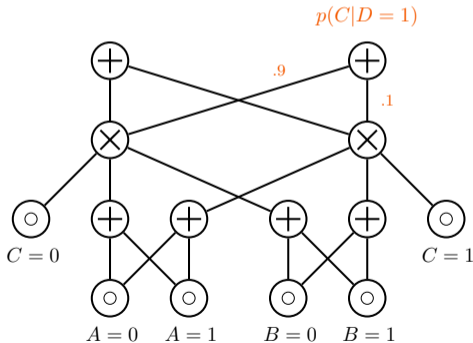
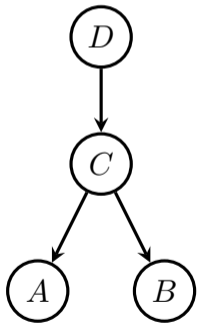
...and recurse over parents...



From BN trees to circuits

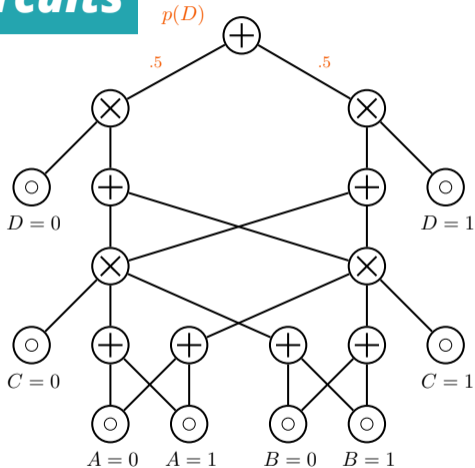
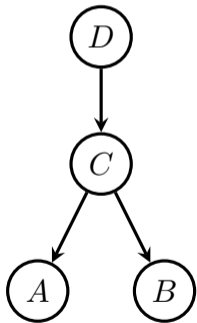
via compilation

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



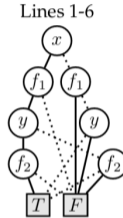
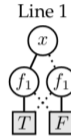
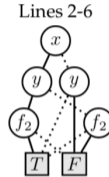
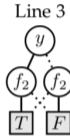
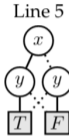
From BN trees to circuits

via compilation



Compilation : probabilistic programming

```
1 x = flip( $\theta_1$ );  
2 if(x) {  
3   y = flip( $\theta_2$ )  
4 } else {  
5   y = x  
6 }
```



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

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

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

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

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

Low-treewidth PGMs

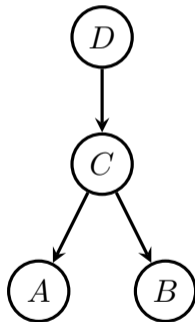
Tree, polytrees and
Thin Junction trees
can be turned into

- decomposable
- smooth
- deterministic

circuits

Therefore they support
tractable

- EVI
- MAR/CON
- MAP



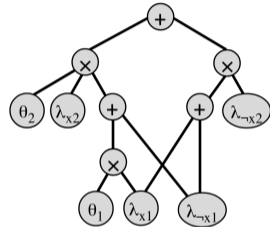
Arithmetic Circuits (ACs)

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]

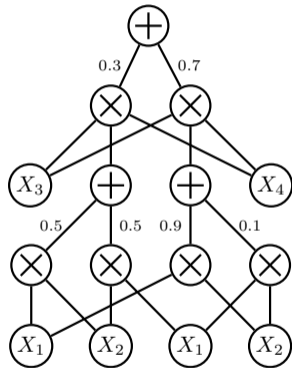
Sum-Product Networks (SPNs)

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. 2014]

Cutset Networks (C Nets)

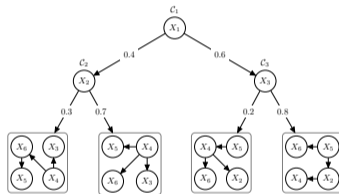
C Nets

[Rahman et al. 2014] are

- decomposable
- smooth
- deterministic

They support tractable

- EVI
- MAR/CON
- MAP



Rahman et al., "Cutset Networks: A Simple, Tractable, and Scalable Approach for Improving the Accuracy of Chow-Liu Trees", 2014

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

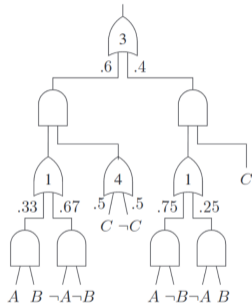
Probabilistic Sentential Decision Diagrams

PSDDs [Kisa et al. 2014b] 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

AndOrGraphs

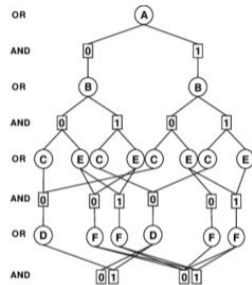
AndOrGarphs

[Dechter et al. 2007] are

- structured
- decomposable
- smooth
- deterministic

They support tractable

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



Dechter et al., "AND/OR search spaces for graphical models", 2007

Marinescu et al., "Best-first AND/OR search for 0/1 integer programming", 2007

Smooth ∨ **decomposable** ∨ **deterministic**

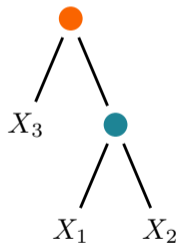
∨ **structured decomposable** **PCs?**

	<i>smooth</i>	<i>dec.</i>	<i>det.</i>	<i>str.dec.</i>
Arithmetic Circuits (ACs) [Darwiche 2003]	✓	✓	✓	✗
Sum-Product Networks (SPNs) [Poon et al. 2011]	✓	✓	✗	✗
Cutset Networks (CNets) [Rahman et al. 2014]	✓	✓	✓	✗
PSDDs [Kisa et al. 2014b]	✓	✓	✓	✓
AndOrGraphs [Dechter et al. 2007]	✓	✓	✓	✓

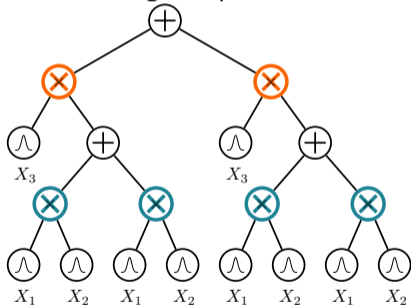
Structured decomposability

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

\Rightarrow stronger requirement than decomposability



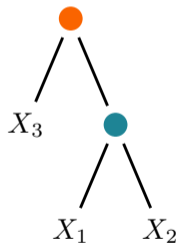
vtree



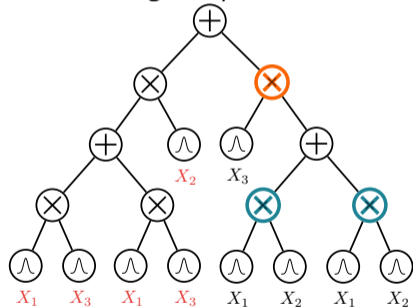
structured decomposable circuit

Structured decomposability

A product node is structured decomposable if decomposes according to a node in a **vtree**
 \Rightarrow stronger requirement than decomposability



vtree



non structured decomposable circuit

Probability of logical events

q₈: *What is the probability of having a traffic jam on my route to work?*



© fineartamerica.com

Probability of logical events

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

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

\Rightarrow *marginals + logical events*



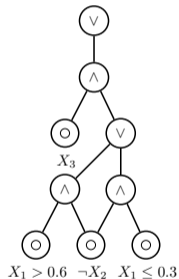
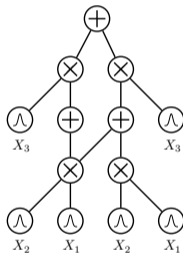
© fineartamerica.com

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

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

Multilinear in circuit sizes if the logical circuit:

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



Smoothness + structured decomp. = tractable PR

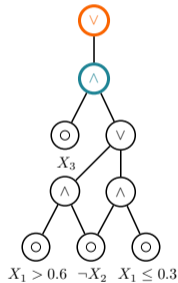
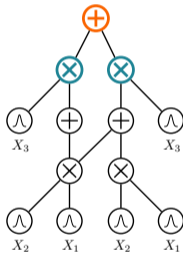
If $p(\mathbf{x}) = \sum_i w_i p_i(\mathbf{x})$, $\alpha = \bigvee_j \alpha_j$,

(smooth p)

(smooth + deterministic α):

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

\Rightarrow probabilities are "pushed down" to children

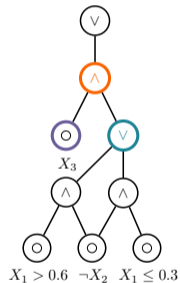
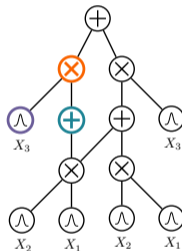


Smoothness + structured decomp. = tractable PR

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

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

\Rightarrow probabilities decompose into simpler ones



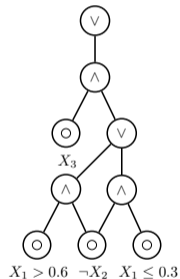
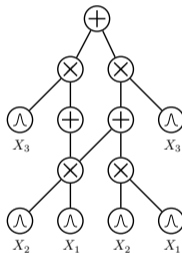
Smoothness + **structured decomp.** = **tractable PR**

To compute $p(\alpha)$:

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

⇒ *cache the values!*

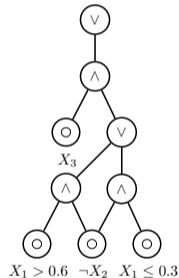
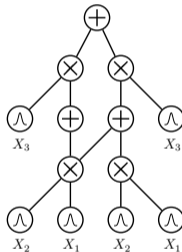
- feedforward evaluation (bottom-up)



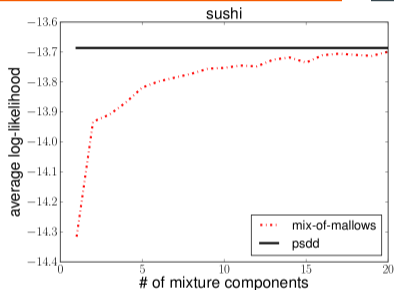
Smoothness + **structured decomp.** = **tractable PR**

To compute $p(\alpha)$:

- compute the probability for each **pair** of probabilistic and logical circuit nodes for the **same vtree node**
- \Rightarrow *cache the values!*
- feedforward evaluation (bottom-up)



ADV inference : preference learning



Preferences and rankings as logical constraints

Structured decomposable circuits for inference over structured spaces

SOTA on modeling densities over rankings

Choi et al., "Tractable learning for structured probability spaces: A case study in learning preference distributions", 2015

Shen et al., "A Tractable Probabilistic Model for Subset Selection.", 2017

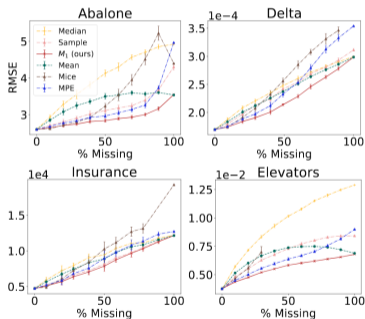
structured decomposability = **tractable...**

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

For the “right” vtree

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

ADV inference : expected predictions



Reasoning about the output of a classifier or regressor f given a distribution p over the input features

\Rightarrow missing values at test time
 \Rightarrow exploratory classifier analysis

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

Closed form moments for f and p as structured decomposable circuits with same v-tree

Stay tuned for...

Next:

1. *How precise is the characterization of tractable circuits by structural properties?* \Rightarrow *necessary conditions*

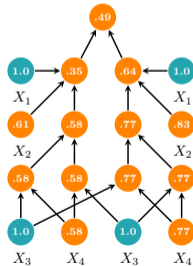
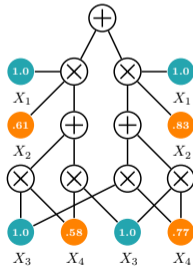
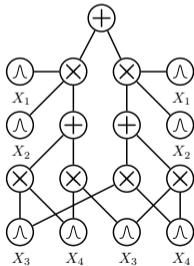
2. *How do structural constraints affect the circuit sizes?* \Rightarrow *succinctness analysis*

After:

Conclusions!

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

Recall: Smoothness and decomposability are **sufficient** conditions for partial evidence evaluation of a circuit to compute marginals.



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

*Smoothness and decomposability are **necessary** and **sufficient** conditions for partial evidence evaluation of a circuit to compute marginals.*

- *Non-smooth node \Rightarrow a variable is unaccounted for \Rightarrow missing integrals.*
- *Non-decomposable node \Rightarrow integral does not decompose.*

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

*Smoothness and decomposability are **necessary** and **sufficient** conditions for partial evidence evaluation of a circuit to compute marginals.*

- *Non-smooth node \Rightarrow a variable is unaccounted for \Rightarrow missing integrals.*
- *Non-decomposable node \Rightarrow integral does not decompose.*

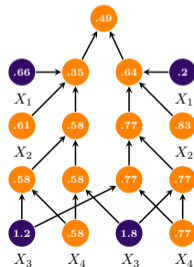
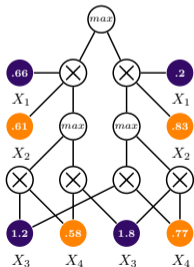
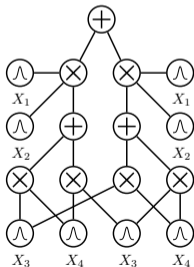
Smoothness + **decomposability** = **tractable MAR**

*Smoothness and decomposability are **necessary** and **sufficient** conditions for partial evidence evaluation of a circuit to compute marginals.*

- *Non-smooth node \Rightarrow a variable is unaccounted for \Rightarrow missing integrals.*
- *Non-decomposable node \Rightarrow integral does not decompose.*

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

Recall: Determinism and decomposability are **sufficient** conditions for maximizer circuit evaluation to compute MAP.



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

Recall: Determinism and decomposability are **sufficient** conditions for maximizer circuit evaluation to compute MAP.

Decomposability is not necessary!

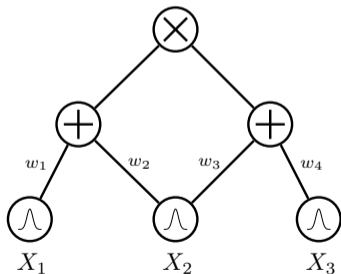


A weaker condition, **consistency**, suffices.

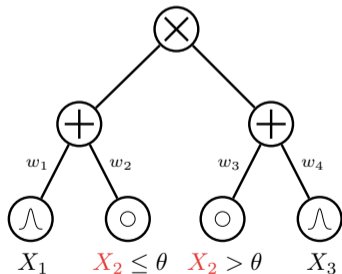
Consistency

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

\Rightarrow decomposability implies consistency



consistent circuit



inconsistent circuit

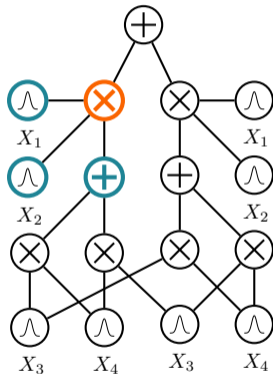
Determinism + ***consistency*** = ***tractable MAP***

Determinism + consistency = tractable MAP

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

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

\Rightarrow solving optimization independently



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

*Determinism and consistency are **necessary** and **sufficient** conditions for maximizer circuit evaluation to compute MAP.*

- *Non-deterministic node \Rightarrow cannot maximize correctly without summations.*
- *Inconsistent node \Rightarrow MAP assignments of children conflict with each other.*

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

*Determinism and consistency are **necessary** and **sufficient** conditions for maximizer circuit evaluation to compute MAP.*

- *Non-deterministic node \Rightarrow cannot maximize correctly without summations.*
- *Inconsistent node \Rightarrow MAP assignments of children conflict with each other.*

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

*Determinism and consistency are **necessary** and **sufficient** conditions for maximizer circuit evaluation to compute MAP.*

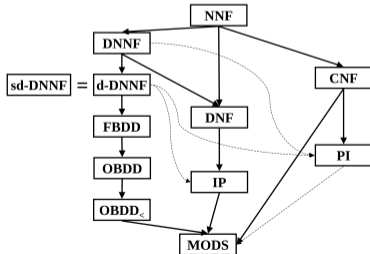
- *Non-deterministic node \Rightarrow cannot maximize correctly without summations.*
- *Inconsistent node \Rightarrow MAP assignments of children conflict with each other.*

Expressive efficiency of circuits

Tractability is defined w.r.t. the size of the model.

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

⇒ Again, connections to logical circuits



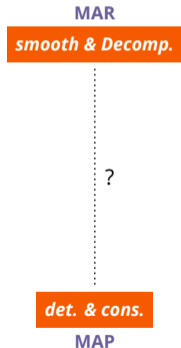
Expressive efficiency of circuits

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

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

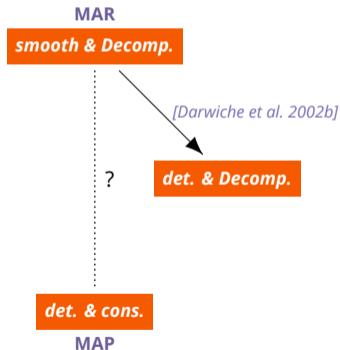
\Rightarrow strictly more succinct iff $\mathcal{M}_1 \leq \mathcal{M}_2$ and $\mathcal{M}_1 \not\leq \mathcal{M}_2$

Expressive efficiency of circuits



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

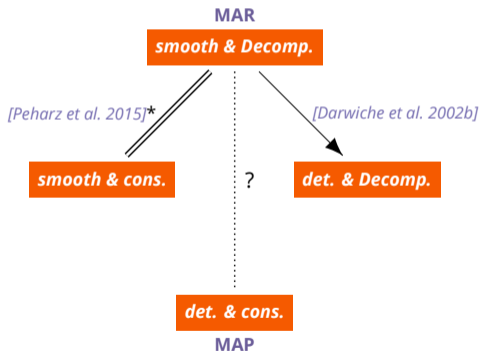
Expressive efficiency of circuits



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

→ : strictly more succinct

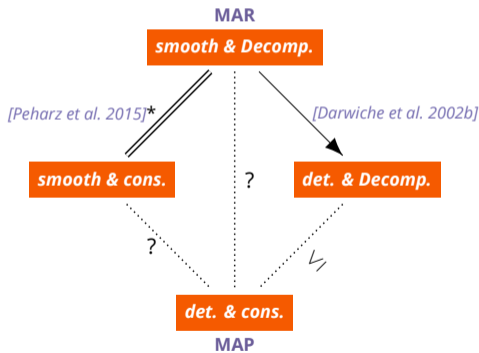
Expressive efficiency of circuits



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

→ : strictly more succinct
== : equally succinct

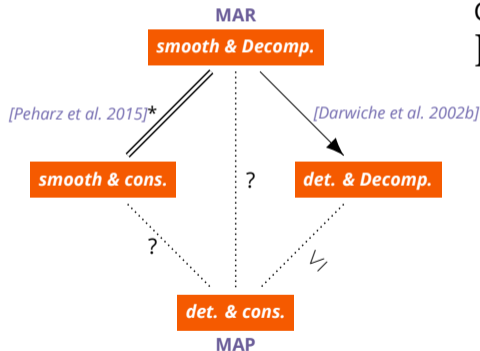
Expressive efficiency of circuits



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

—▶ : strictly more succinct
== : equally succinct

Expressive efficiency of circuits



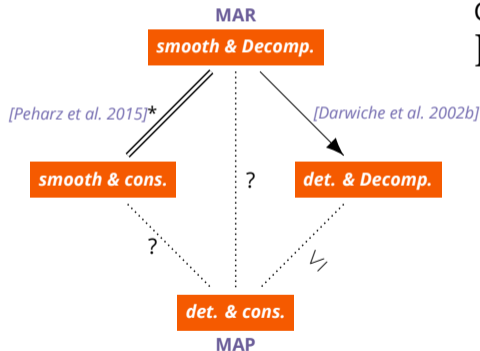
→ : strictly more succinct
== : equally succinct

Consider following circuit over Boolean variables:

$$\prod_i^r (Y_i \cdot Z_{i1} + (\neg Y_i) \cdot Z_{i2}), \quad Z_{ij} \in \mathbf{X}$$

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

Expressive efficiency of circuits



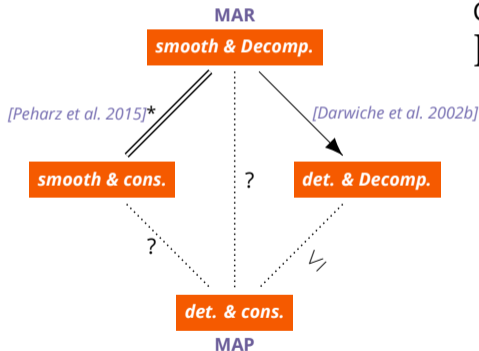
—▶ : strictly more succinct
== : equally succinct

Consider following circuit over Boolean variables:

$$\prod_i^r (Y_i \cdot Z_{i1} + (\neg Y_i) \cdot Z_{i2}), \quad Z_{ij} \in \mathbf{X}$$

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

Expressive efficiency of circuits



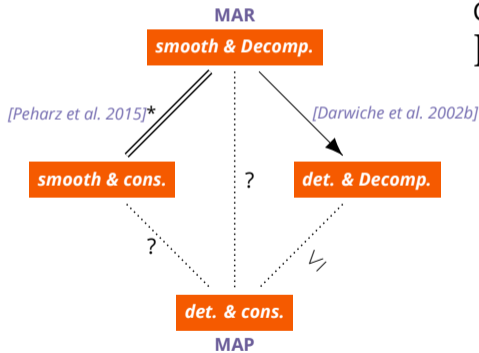
→ : strictly more succinct
== : equally succinct

Consider following circuit over Boolean variables:

$$\prod_i^r (Y_i \cdot Z_{i1} + (\neg Y_i) \cdot Z_{i2}), \quad Z_{ij} \in \mathbf{X}$$

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

Expressive efficiency of circuits



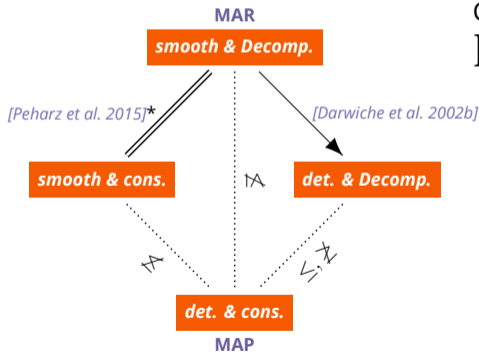
→ : strictly more succinct
== : equally succinct

Consider following circuit over Boolean variables:

$$\prod_i^r (Y_i \cdot Z_{i1} + (\neg Y_i) \cdot Z_{i2}), \quad Z_{ij} \in \mathbf{X}$$

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

Expressive efficiency of circuits



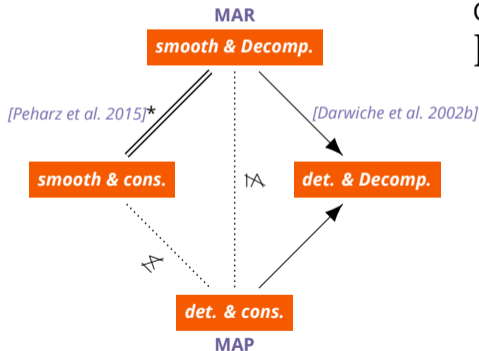
—▶ : strictly more succint
== : equally succint

Consider following circuit over Boolean variables:

$$\prod_i^r (Y_i \cdot Z_{i1} + (\neg Y_i) \cdot Z_{i2}), \quad Z_{ij} \in \mathbf{X}$$

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

Expressive efficiency of circuits



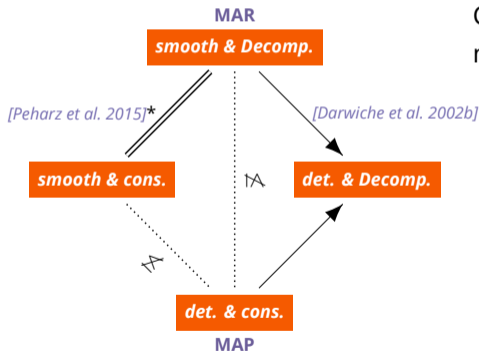
—▶ : strictly more succinct
== : equally succinct

Consider following circuit over Boolean variables:

$$\prod_i^r (Y_i \cdot Z_{i1} + (\neg Y_i) \cdot Z_{i2}), \quad Z_{ij} \in \mathbf{X}$$

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

Expressive efficiency of circuits

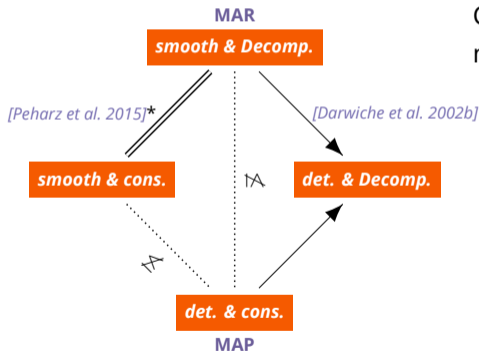


—▶ : strictly more succinct
== : equally succinct

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

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

Expressive efficiency of circuits

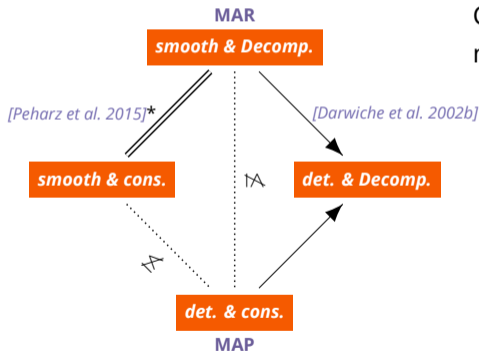


Consider the marginal distribution $p(\mathbf{X})$ from a naive Bayes distribution $p(\mathbf{X}, C)$:

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

—▶ : strictly more succinct
== : equally succinct

Expressive efficiency of circuits

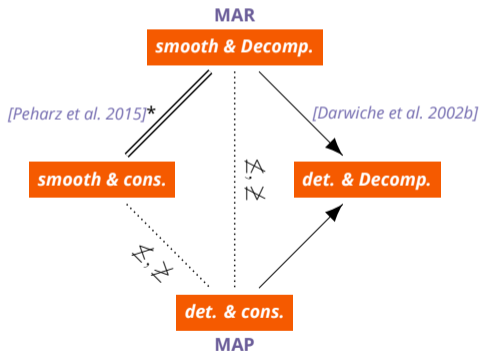


—▶ : strictly more succinct
== : equally succinct

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

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

Expressive efficiency of circuits



—▶ : strictly more succinct
== : equally succinct

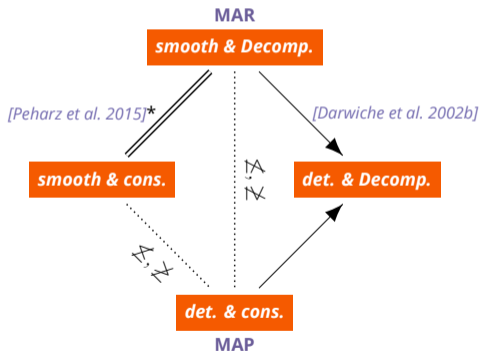
Neither smooth & decomposable nor deterministic & consistent circuits are more succinct than the other!

⇒ Choose tractable circuit family based on your query

More theoretical questions remaining

⇒ "Complete the map"

Expressive efficiency of circuits



—▶ : strictly more succinct
== : equally succinct

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

⇒ Choose tractable circuit family based on your query

■ More theoretical questions remaining

⇒ "Complete the map"

Conclusions

Why tractable inference?

or expressiveness vs tractability

Probabilistic circuits

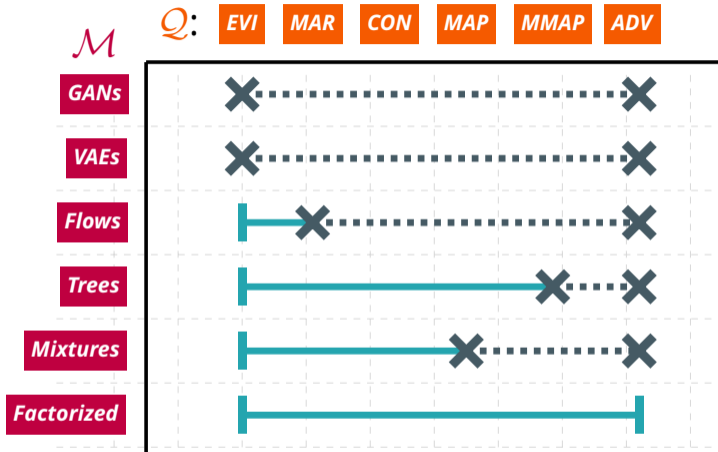
a unified framework for tractable probabilistic modeling

Learning circuits

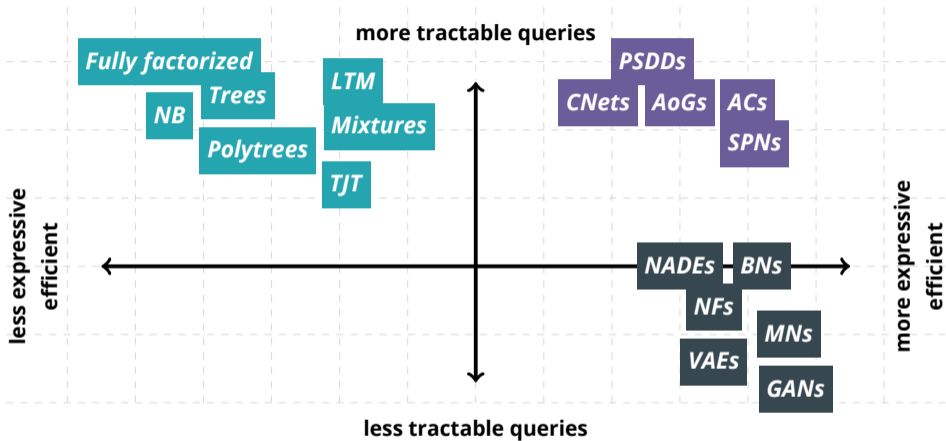
learning their structure and parameters from data

Representations and theory

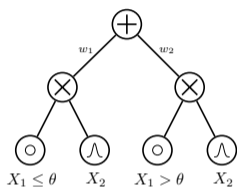
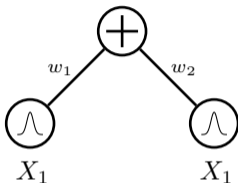
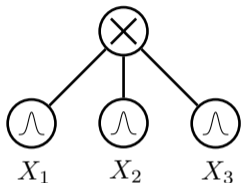
tracing the boundaries of tractability and connections to other formalisms



takeaway #1: tractability is a spectrum



takeaway #2: you can be both tractable and expressive



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

Challenge #1

hybridizing tractable and intractable models

Hybridize probabilistic inference:

tractable models inside intractable loops

and intractable small boxes glued by tractable inference!

Challenge #2

scaling tractable learning

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

Challenge #3

deep theoretical understanding

Trace a precise picture
of the ***whole tractable spectrum***
and ***complete the map of succinctness!***

Challenge #4

advanced and automated reasoning

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

Readings

Probabilistic circuits: Representation and Learning

`starai.cs.ucla.edu/papers/LecNoAAAI20.pdf`

Foundations of Sum-Product Networks for probabilistic modeling

`tinyurl.com/w65po5d`

Slides for this tutorial

`starai.cs.ucla.edu/slides/AAAI20.pdf`

Code

Juice.jl advanced logical+probabilistic inference with circuits in Julia

github.com/Juice-jl/ProbabilisticCircuits.jl

SumProductNetworks.jl SPN routines in Julia

github.com/trappmartin/SumProductNetworks.jl

SPFlow easy and extensible python library for SPNs

github.com/SPFlow/SPFlow

Libra several structure learning algorithms in OCaml

libra.cs.uoregon.edu

More refs \Rightarrow github.com/arranger1044/awesome-spn

Acknowledgments

We thank **Nicola Di Mauro** for his help with a previous version of this tutorial we gave at UAI19:

<http://web.cs.ucla.edu/~guyvdb/slides/TPMTutorialUAI19.pdf>

In the same way,

we are grateful to all the people from the logical and probabilistic circuits communities whose insightful discussions, ideas and research helped and inspired us.

(12 pages of references incoming!)

References I

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

References II

- ⊕ Yedidia, Jonathan S, William T Freeman, and Yair Weiss (2001). "Generalized belief propagation". In: *Advances in neural information processing systems*, pp. 689–695.
- ⊕ Chickering, Max (2002). "The WinMine Toolkit". In: *Microsoft, Redmond*.
- ⊕ Darwiche, Adnan and Pierre Marquis (2002a). "A knowledge compilation map". In: *Journal of Artificial Intelligence Research* 17, pp. 229–264.
- ⊕ — (2002b). "A knowledge compilation map". In: *Journal of Artificial Intelligence Research* 17.1, pp. 229–264.
- ⊕ Dechter, Rina, Kalev Kask, and Robert Mateescu (2002). "Iterative join-graph propagation". In: *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., pp. 128–136.
- ⊕ Darwiche, Adnan (2003). "A Differential Approach to Inference in Bayesian Networks". In: *JACM*.
- ⊕ 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.

References III

- ⊕ Marinescu, Radu and Rina Dechter (2007). "Best-first AND/OR search for 0/1 integer programming". In: *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*. Springer, pp. 171–185.
- ⊕ Riguzzi, Fabrizio (2007). "A top down interpreter for LPAD and CP-logic". In: *Congress of the Italian Association for Artificial Intelligence*. Springer, pp. 109–120.
- ⊕ Lowd, Daniel and Pedro Domingos (2008). "Learning Arithmetic Circuits". In: *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*. UAI'08. Helsinki, Finland: AUAI Press, pp. 383–392. ISBN: 0-9749039-4-9. URL: <http://dl.acm.org/citation.cfm?id=3023476.3023522>.
- ⊕ Olteanu, Dan and Jiewen Huang (2008). "Using OBDDs for efficient query evaluation on probabilistic databases". In: *International Conference on Scalable Uncertainty Management*. Springer, pp. 326–340.
- ⊕ Koller, Daphne and Nir Friedman (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.
- ⊕ Choi, Arthur and Adnan Darwiche (2010). "Relax, compensate and then recover". In: *JSAI International Symposium on Artificial Intelligence*. Springer, pp. 167–180.
- ⊕ Lowd, Daniel and Pedro Domingos (2010). "Approximate inference by compilation to arithmetic circuits". In: *Advances in Neural Information Processing Systems*, pp. 1477–1485.
- ⊕ Campos, Cassio Polpo de (2011). "New complexity results for MAP in Bayesian networks". In: *IJCAI*. Vol. 11, pp. 2100–2106.
- ⊕ Darwiche, Adnan (2011). "SDD: A New Canonical Representation of Propositional Knowledge Bases". In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*. IJCAI'11. Barcelona, Catalonia, Spain. ISBN: 978-1-57735-514-4.
- ⊕ de Campos, Cassio P (2011). "New complexity results for MAP in Bayesian networks". In: *IJCAI*. Vol. 11, pp. 2100–2106.
- ⊕ Larochelle, Hugo and Iain Murray (2011). "The Neural Autoregressive Distribution Estimator". In: *International Conference on Artificial Intelligence and Statistics*, pp. 29–37.

References IV

- ⊕ Poon, Hoifung and Pedro Domingos (2011). “Sum-Product Networks: a New Deep Architecture”. In: *UAI 2011*.
- ⊕ Sontag, David, Amir Globerson, and Tommi Jaakkola (2011). “Introduction to dual decomposition for inference”. In: *Optimization for Machine Learning 1*, pp. 219–254.
- ⊕ Gens, Robert and Pedro Domingos (2012). “Discriminative Learning of Sum-Product Networks”. In: *Advances in Neural Information Processing Systems 25*, pp. 3239–3247.
- ⊕ — (2013). “Learning the Structure of Sum-Product Networks”. In: *Proceedings of the ICML 2013*, pp. 873–880.
- ⊕ 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*.
- ⊕ Goodfellow, Ian et al. (2014). “Generative adversarial nets”. In: *Advances in neural information processing systems*, pp. 2672–2680.
- ⊕ Kingma, Diederik P and Max Welling (2014). “Auto-Encoding Variational Bayes”. In: *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*. 2014.
- ⊕ Kisa, Doga et al. (2014a). “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>.
- ⊕ — (2014b). “Probabilistic sentential decision diagrams”. In: *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*. Vienna, Austria.
- ⊕ Martens, James and Venkatesh Medabalimi (2014). “On the Expressive Efficiency of Sum Product Networks”. In: *CoRR* abs/1411.7717.

References V

- ⊕ Peharz, Robert, Robert Gens, and Pedro Domingos (2014). "Learning Selective Sum-Product Networks". In: *Workshop on Learning Tractable Probabilistic Models*. LTPM.
- ⊕ Rahman, Tahrima, Prasanna Kothalkar, and Vibhav Gogate (2014). "Cutset Networks: A Simple, Tractable, and Scalable Approach for Improving the Accuracy of Chow-Liu Trees". In: *Machine Learning and Knowledge Discovery in Databases*. Vol. 8725. LNCS. Springer, pp. 630-645.
- ⊕ Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic backprop. and approximate inference in deep generative models". In: *arXiv preprint arXiv:1401.4082*.
- ⊕ Rooshenas, Amirmohammad and Daniel Lowd (2014). "Learning Sum-Product Networks with Direct and Indirect Variable Interactions". In: *Proceedings of ICML 2014*.
- ⊕ Adel, Tameem, David Balduzzi, and Ali Ghodsi (2015). "Learning the Structure of Sum-Product Networks via an SVD-based Algorithm". In: *Uncertainty in Artificial Intelligence*.
- ⊕ Bekker, Jessa et al. (2015). "Tractable Learning for Complex Probability Queries". In: *Advances in Neural Information Processing Systems 28 (NIPS)*.
- ⊕ Burda, Yuri, Roger Grosse, and Ruslan Salakhutdinov (2015). "Importance weighted autoencoders". In: *arXiv preprint arXiv:1509.00519*.
- ⊕ Choi, Arthur, Guy Van den Broeck, and Adnan Darwiche (2015a). "Tractable learning for structured probability spaces: A case study in learning preference distributions". In: *Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*.
- ⊕ Choi, Arthur, Guy Van Den Broeck, and Adnan Darwiche (2015b). "Tractable Learning for Structured Probability Spaces: A Case Study in Learning Preference Distributions". In: *Proceedings of the 24th International Conference on Artificial Intelligence*. IJCAI'15. Buenos Aires, Argentina: AAAI Press, pp. 2861-2868. ISBN: 978-1-57735-738-4. URL: <http://dl.acm.org/citation.cfm?id=2832581.2832649>.
- ⊕ Dennis, Aaron and Dan Ventura (2015). "Greedy Structure Search for Sum-product Networks". In: *IJCAI'15*. Buenos Aires, Argentina: AAAI Press, pp. 932-938. ISBN: 978-1-57735-738-4.

References VI

- ⊕ Di Mauro, Nicola, Antonio Vergari, and Floriana Esposito (2015a). "Learning Accurate Cutset Networks by Exploiting Decomposability". In: *Proceedings of AIXIA*. Springer, pp. 221–232.
- ⊕ Di Mauro, Nicola, Antonio Vergari, and Teresa M.A. Basile (2015b). "Learning Bayesian Random Cutset Forests". In: *Proceedings of ISMIS*. Springer, pp. 122–132.
- ⊕ Fierens, Daan et al. (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/FierensTLP15.pdf>.
- ⊕ Germain, Mathieu et al. (2015). "MADE: Masked Autoencoder for Distribution Estimation". In: *CoRR abs/1502.03509*.
- ⊕ Peharz, Robert (2015). "Foundations of Sum-Product Networks for Probabilistic Modeling". PhD thesis. Graz University of Technology, SPSC.
- ⊕ Peharz, Robert et al. (2015). "On Theoretical Properties of Sum-Product Networks". In: *The Journal of Machine Learning Research*.
- ⊕ Vergari, Antonio, Nicola Di Mauro, and Floriana Esposito (2015). "Simplifying, Regularizing and Strengthening Sum-Product Network Structure Learning". In: *ECML-PKDD 2015*.
- ⊕ Vlasselaeer, 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/VlasselaeerIJCAI15.pdf>.
- ⊕ Belle, Vaishak and Luc De Raedt (2016). "Semiring Programming: A Framework for Search, Inference and Learning". In: *arXiv preprint arXiv:1609.06954*.
- ⊕ Cohen, Nadav, Or Sharir, and Amnon Shashua (2016). "On the expressive power of deep learning: A tensor analysis". In: *Conference on Learning Theory*, pp. 698–728.
- ⊕ Friesen, Abram L and Pedro Domingos (2016). "Submodular Sum-product Networks for Scene Understanding". In:

References VII

- ⊕ 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>.
- ⊕ Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu (2016). "Pixel recurrent neural networks". In: *arXiv preprint arXiv:1601.06759*.
- ⊕ Oztok, Umut, Arthur Choi, and Adnan Darwiche (2016). "Solving PP-PP-complete problems using knowledge compilation". In: *Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning*.
- ⊕ Peharz, Robert et al. (2016). "On the Latent Variable Interpretation in Sum-Product Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP, Issue 99. URL: <http://arxiv.org/abs/1601.06180>.
- ⊕ Pronobis, A. and R. P. N. Rao (2016). "Learning Deep Generative Spatial Models for Mobile Robots". In: *ArXiv e-prints*. arXiv: 1610.02627 [cs.RD].
- ⊕ Rahman, Tahrira and Vibhav Gogate (2016). "Learning Ensembles of Cutset Networks". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI'16. Phoenix, Arizona: AAAI Press, pp. 3301–3307. URL: <http://dl.acm.org/citation.cfm?id=3016100.3016365>.
- ⊕ Rashwan, Abdullah, Han Zhao, and Pascal Poupart (2016). "Online and Distributed Bayesian Moment Matching for Parameter Learning in Sum-Product Networks". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pp. 1469–1477.
- ⊕ Rooshenas, Amirmohammad and Daniel Lowd (2016). "Discriminative Structure Learning of Arithmetic Circuits". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pp. 1506–1514.
- ⊕ Sguerra, Bruno Massoni and Fabio G Cozman (2016). "Image classification using sum-product networks for autonomous flight of micro aerial vehicles". In: *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*. IEEE, pp. 139–144.

References VIII

- ⊕ Sharir, Or et al. (2016). "Tractable generative convolutional arithmetic circuits". In: *arXiv preprint arXiv:1610.04167*.
- ⊕ Shen, Yujia, Arthur Choi, and Adnan Darwiche (2016). "Tractable Operations for Arithmetic Circuits of Probabilistic Models". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3936–3944.
- ⊕ Vlasselaer, Jonas et al. (2016). "Exploiting Local and Repeated Structure in Dynamic Bayesian Networks". In: *Artificial Intelligence* 232, pp. 43–53. ISSN: 0004-3702. DOI: 10.1016/j.artint.2015.12.001.
- ⊕ Yuan, Zehuan 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*.
- ⊕ Choi, YooJung, Adnan Darwiche, and Guy Van den Broeck (2017). "Optimal feature selection for decision robustness in Bayesian networks". In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*.
- ⊕ Conaty, Diarmaid, Denis Deratani Mauá, and Cassio Polpo de Campos (2017). "Approximation Complexity of Maximum A Posteriori Inference in Sum-Product Networks". In: *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence*. Ed. by Gal Elidan and Kristian Kersting. AUAI Press, pp. 322–331.
- ⊕ Di Mauro, Nicola et al. (2017). "Fast and Accurate Density Estimation with Extremely Randomized Cutset Networks". In: *ECML-PKDD 2017*.

References IX

- ⊕ Kimmig, Angelika, Guy Van den Broeck, and Luc De Raedt (2017). "Algebraic model counting". In: *Journal of Applied Logic* 22, pp. 46–62.
- ⊕ Liang, Yitao, Jessa Bekker, and Guy Van den Broeck (2017a). "Learning the structure of probabilistic sentential decision diagrams". In: *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*.
- ⊕ Liang, Yitao and Guy Van den Broeck (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.
- ⊕ Salimans, Tim et al. (2017). "PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications". In: *arXiv preprint arXiv:1701.05517*.
- ⊕ Shen, Yujia, Arthur Choi, and Adnan Darwiche (2017). "A Tractable Probabilistic Model for Subset Selection.". In: *UAI*.
- ⊕ Van den Broeck, Guy and Dan Suciu (2017). *Query Processing on Probabilistic Data: A Survey*. Foundations and Trends in Databases. Now Publishers. DOI: 10.1561/19000000052. URL: <http://starai.cs.ucla.edu/papers/VdBFTDB17.pdf>.
- ⊕ Butz, Cory J et al. (2018). "An Empirical Study of Methods for SPN Learning and Inference". In: *International Conference on Probabilistic Graphical Models*, pp. 49–60.
- ⊕ Choi, Yoojung and Guy Van den Broeck (2018). "On robust trimming of Bayesian network classifiers". In: *arXiv preprint arXiv:1805.11243*.

References X

- ⊕ 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 (2018). "Approximate Knowledge Compilation by Online Collapsed Importance Sampling". In: *Advances in Neural Information Processing Systems 31 (NeurIPS)*. URL: <http://starai.cs.ucla.edu/papers/FriedmanNeurIPS18.pdf>.
- ⊕ Jaini, Priyank, Amur Ghose, and Pascal Poupart (2018). "Prometheus: Directly Learning Acyclic Directed Graph Structures for Sum-Product Networks". In: *International Conference on Probabilistic Graphical Models*, pp. 181–192.
- ⊕ Molina, Alejandro et al. (2018). "Mixed Sum-Product Networks: A Deep Architecture for Hybrid Domains". In: *AAAI*.
- ⊕ 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.
- ⊕ Shen, Yujia, Arthur Choi, and Adnan Darwiche (2018). "Conditional PSDDs: Modeling and learning with modular knowledge". In: *Thirty-Second AAAI Conference on Artificial Intelligence*.
- ⊕ 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*.
- ⊕ Dai, Bin and David Wipf (2019). "Diagnosing and enhancing vae models". In: *arXiv preprint arXiv:1903.05789*.
- ⊕ Ghosh, Partha et al. (2019). "From variational to deterministic autoencoders". In: *arXiv preprint arXiv:1903.12436*.
- ⊕ Holtzen, Steven, Todd Millstein, and Guy Van den Broeck (2019). "Symbolic Exact Inference for Discrete Probabilistic Programs". In: *arXiv preprint arXiv:1904.02079*.
- ⊕ Khosravi, Pasha et al. (2019a). "On Tractable Computation of Expected Predictions". In: *Advances in Neural Information Processing Systems*, pp. 11167–11178.

References XI

- ⊕ Khosravi, Pasha et al. (2019b). “What to Expect of Classifiers? Reasoning about Logistic Regression with Missing Features”. In: *arXiv preprint arXiv:1903.01620*.
- ⊕ Khosravi, Pasha et al. (2019c). “What to Expect of Classifiers? Reasoning about Logistic Regression with Missing Features”. In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*.
- ⊕ Kossen, Jannik et al. (2019). “Structured Object-Aware Physics Prediction for Video Modeling and Planning”. In: *arXiv preprint arXiv:1910.02425*.
- ⊕ Liang, Yitao and Guy Van den Broeck (2019). “Learning Logistic Circuits”. In: *Proceedings of the 33rd Conference on Artificial Intelligence (AAAI)*.
- ⊕ Peharz, Robert et al. (2019a). “Random Sum-Product Networks: A Simple and Effective Approach to Probabilistic Deep Learning”. In: *Uncertainty in Artificial Intelligence*.
- ⊕ Peharz, Robert et al. (2019b). “Random sum-product networks: A simple but effective approach to probabilistic deep learning”. In: *Proceedings of UAI*.
- ⊕ Shao, Xiaoting et al. (2019). “Conditional Sum-Product Networks: Imposing Structure on Deep Probabilistic Architectures”. In: *arXiv preprint arXiv:1905.08550*.
- ⊕ 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>.
- ⊕ 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>.

References XII

- ⊕ Trapp, Martin et al. (2019). "Bayesian Learning of Sum-Product Networks". In: *Advances in neural information processing systems (NeurIPS)*.
- ⊕ Vergari, Antonio et al. (2019). "Automatic Bayesian density analysis". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33, pp. 5207-5215.