Probabilistic Circuits

Inference
Representations
Learning
Theory

Antonio Vergari
University of California, Los Angeles

YooJung Choi
University of California, Los Angeles

Robert Peharz
TU Eindhoven

Guy Van den Broeck
University of California, Los Angeles

September 14th, 2020 - Ghent, Belgium - ***ECML-PKDD 2020***

Fully factorized

NaiveBayes AndOrGraphs PDGs

Trees PSDDs CNets LTMs SPNs NADEs

Thin Junction Trees ACs MADEs MAFs VAEs

DPPs FVSBNs TACs IAFs NAFs RAEs

Mixtures BNs NICE FGs GANs

RealNVP MNs

The Alphabet Soup of probabilistic models

Fully factorized

NaiveBayes AndOrGraphs PDGs

Trees PSDDs CNets LTMs SPNs NADEs

Thin Junction Trees ACs MADEs MAFs VAEs

DPPs FVSBNs TACs IAFs NAFs RAEs

Mixtures BNs NICE FGs GANs

RealNVP MNs

Intractable and *tractable* models

Fully factorized

NaiveBayes *AndOrGraphs* *PDGs*

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

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

DPPs *FVSBNs* *TACs* *IAFs* *NAFs* *RAEs*

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

RealNVP *MNs*

tractability is a spectrum

Fully factorized

NaiveBayes *AndOrGraphs* *PDGs*

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

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

DPPs *FVSBNs* *TACs* *IAFs* *NAFs* *RAEs*

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

RealNVP *MNs*

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

Fully factorized

NaiveBayes *AndOrGraphs* *PDGs*

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

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

DPPs *FVSBNs* *TACs* *IAFs* *NAFs* *RAEs*

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

RealNVP *MNs*

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

Advanced representations

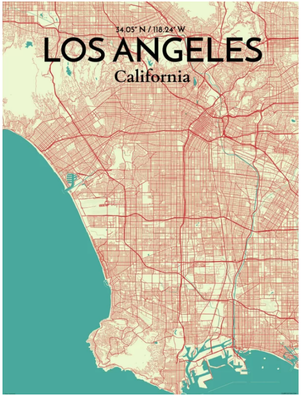
tracing the boundaries of tractability and connections to other formalisms

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 Westwood Blvd.?*

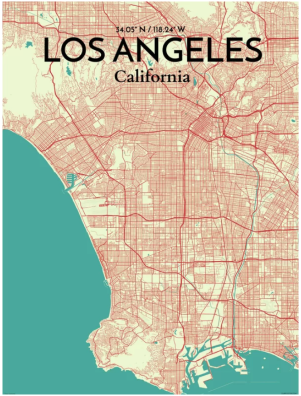


© fineartamerica.com

Why probabilistic inference?

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

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



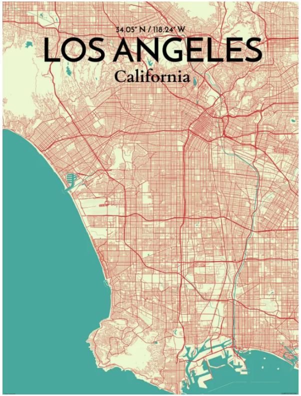
© fineartamerica.com

Why probabilistic inference?

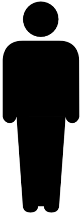
q_1: *What is the probability that today is a Monday and there is a traffic jam on Westwood Blvd.?*

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

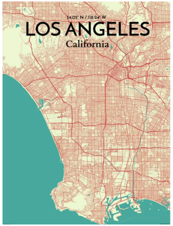
How to answer several of these ***probabilistic queries***?



"What is the most likely street to have a traffic jam at 12.00?"



$\mathbf{q}_1$?

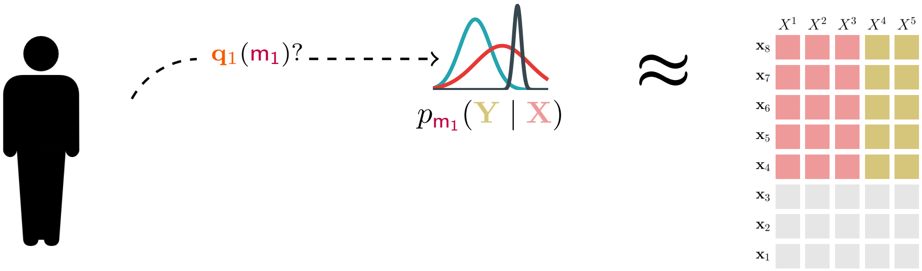


answering queries...

"What is the most likely street to have a traffic jam at 12.00?"

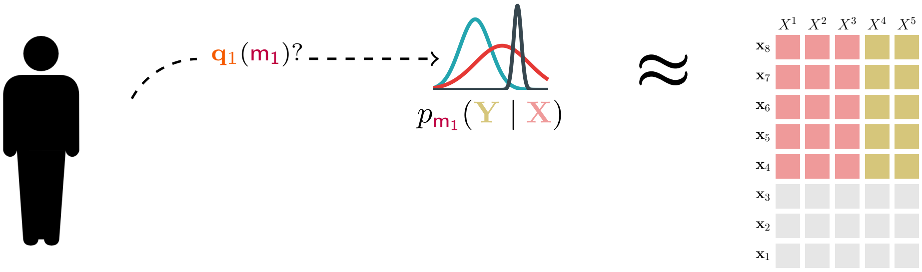
answering queries...

“What is the most likely ***street*** *to have a traffic jam at* ***12.00****?”*



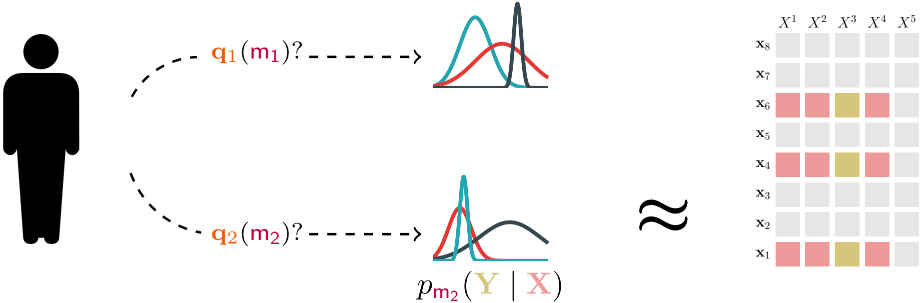
...by fitting predictive models!

“What is the most likely ***street*** *to have a traffic jam at* ***12.00****?”*



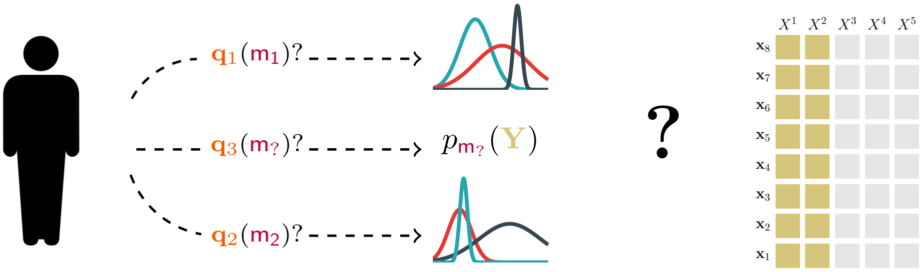
~~***...by fitting predictive models!***~~

*“What is the most likely **time** to see a traffic jam at **Sunset Blvd.**?”*

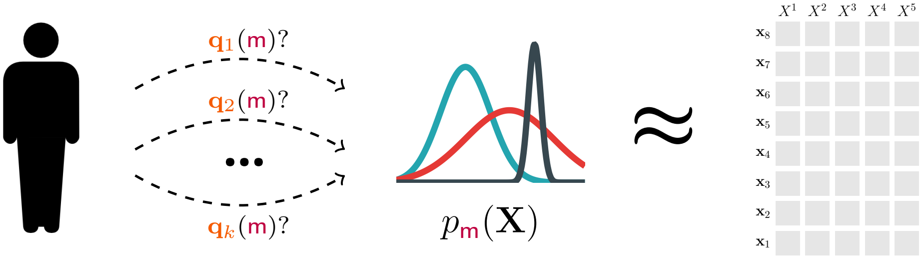


~~***...by fitting predictive models!***~~

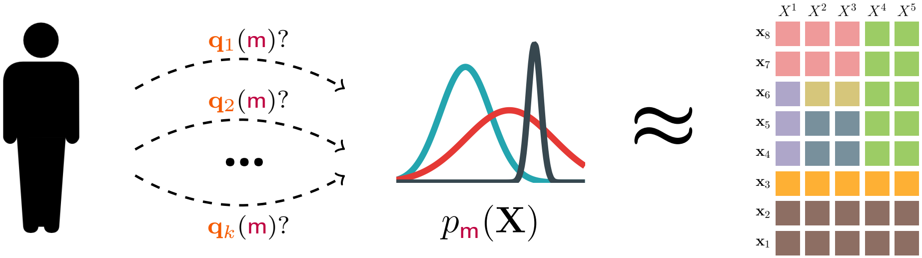
"What is the probability of a traffic jam on ***Westwood Blvd.*** *on* ***Monday****?"*



~~...by fitting predictive models!~~



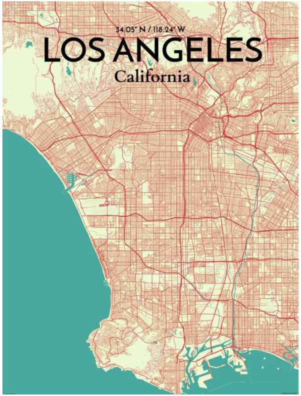
...by fitting generative models!



...e.g. exploratory data analysis

Why probabilistic inference?

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



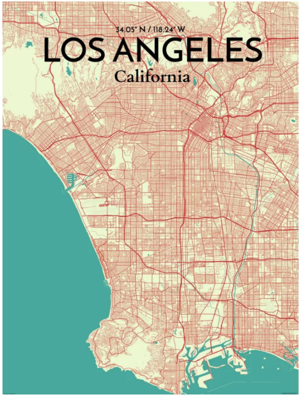
© fineartamerica.com

Why probabilistic inference?

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

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

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



© fineartamerica.com

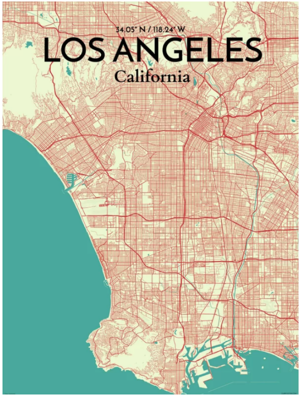
Why probabilistic inference?

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

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

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

$\Longrightarrow$ ***marginals***



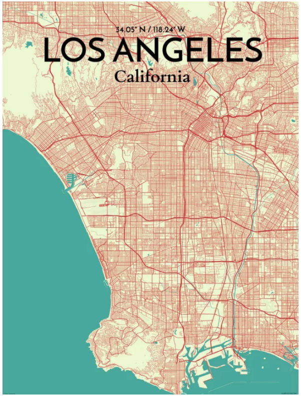
© fineartamerica.com

Why probabilistic inference?

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

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

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



© fineartamerica.com

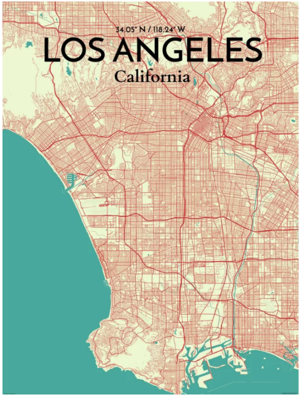
Why probabilistic inference?

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

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

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

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



© fineartamerica.com

Tractable Probabilistic Inference

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

Tractable Probabilistic Inference

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

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

Tractable Probabilistic Inference

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

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

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

Tractable Probabilistic Inference

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

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

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

$\Longrightarrow$ *Why* ***exactness****? Highest guarantee possible!*

Stay tuned for...

Next:

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

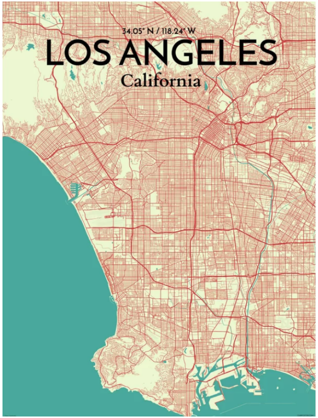
After:

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

tractable bands

Complete evidence (EVI)

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



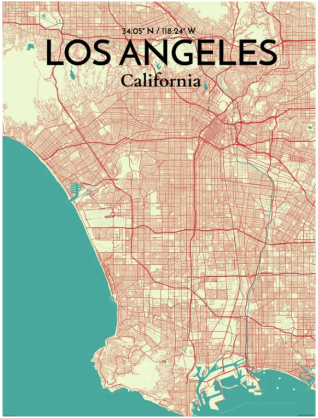
© fineartamerica.com

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 Westwood Blvd.?*

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

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



© fineartamerica.com

Complete evidence (EVI)

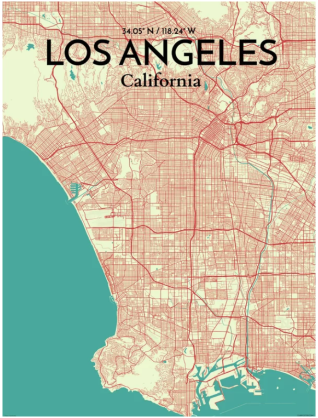
q_3: *What is the probability that today is a Monday at 12.00 and there is a traffic jam only on Westwood Blvd.?*

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

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

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

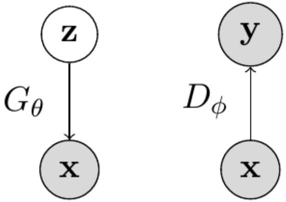
$$\theta_{\mathbf{m}}^{\text{MLE}} = \operatorname{argmax}_\theta \prod_{\mathbf{x} \in \mathcal{D}} p_{\mathbf{m}}(\mathbf{x}; \theta)$$



© fineartamerica.com

Generative Adversarial Networks

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

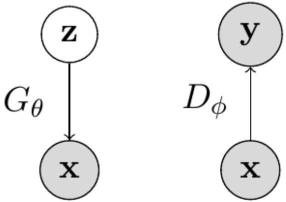


Goodfellow et al., "Generative adversarial nets", 2014

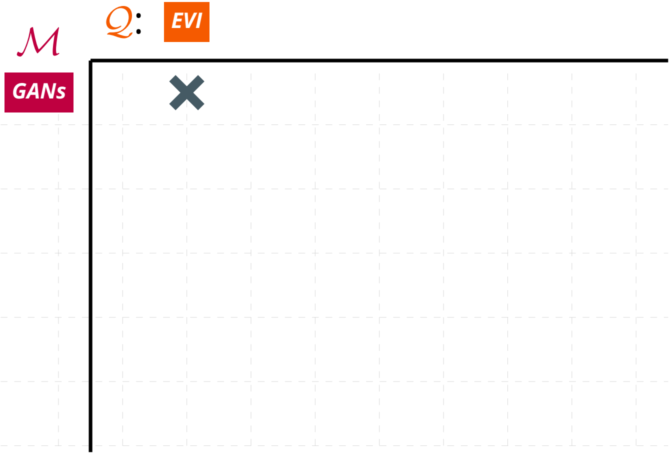
Generative Adversarial Networks

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

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



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

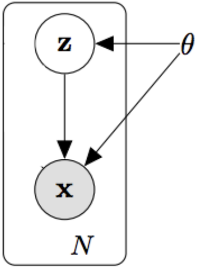


tractable bands

Variational Autoencoders

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

- an explicit likelihood model!



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

~~Variational Autoencoders~~

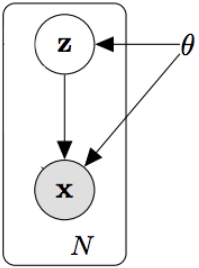
$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x} \mid \mathbf{z})\right] - \mathbb{KL}(q_\phi(\mathbf{z} \mid \mathbf{x})||p(\mathbf{z}))$$

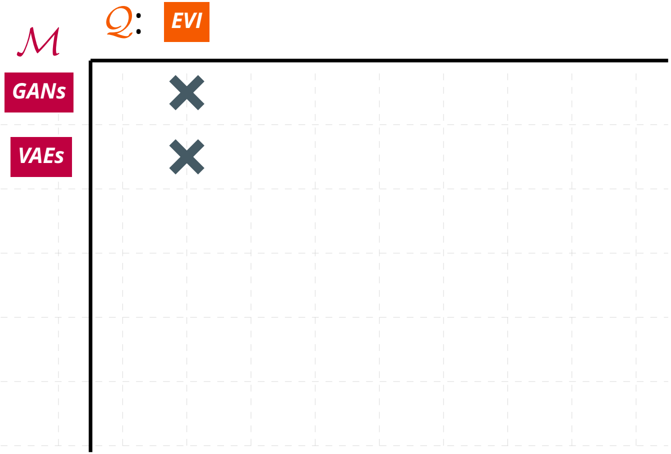
- an explicit likelihood model!
- ... but computing $\log p_\theta(\mathbf{x})$ is intractable

  $\Longrightarrow$ *an infinite and uncountable mixture*

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

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

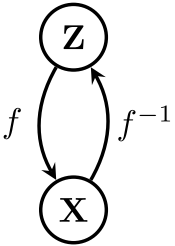




tractable bands

Normalizing flows

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



- an explicit likelihood!
- ...plus structured Jacobians

$\Rightarrow$ *tractable EVI queries!*

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

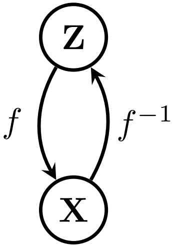
Normalizing flows

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

- an explicit likelihood!
- ...plus structured Jacobians

$\Longrightarrow$ *tractable EVI queries!*

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



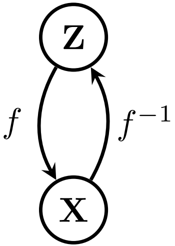
Normalizing flows

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

- an explicit likelihood!
- ...plus structured Jacobians

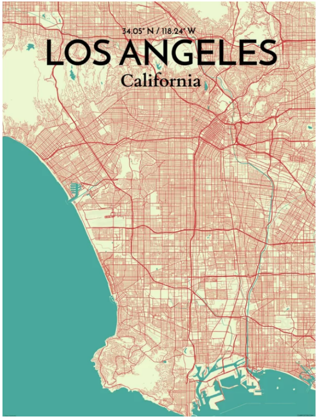
$\Rightarrow$ *tractable EVI queries!*

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



Marginal queries (MAR)

q_1: *What is the probability that today is a Monday ~~at 12.00~~ and there is a traffic jam ~~only~~ on Westwood Blvd.?*

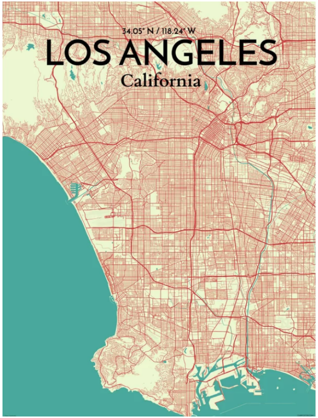


© fineartamerica.com

Marginal queries (MAR)

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

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



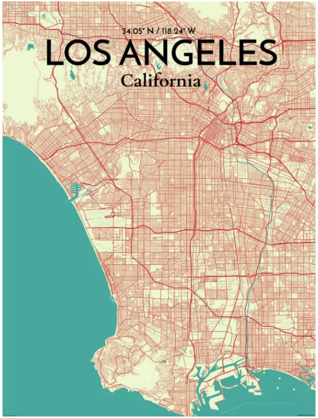
Marginal queries (MAR)

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

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

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

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



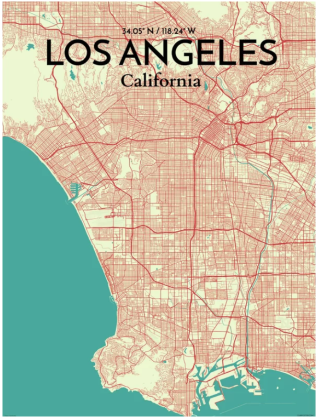
Marginal queries (MAR)

q_1: *What is the probability that today is a Monday ~~at~~ ~~12.00~~ and there is a traffic jam ~~only~~ on Westwood Blvd.?*

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

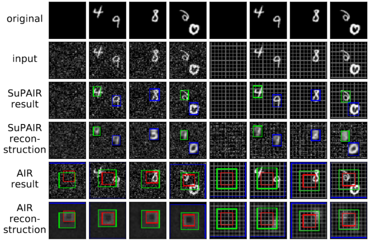
tractable MAR $\Longrightarrow$ tractable ***conditional queries*** (CON):

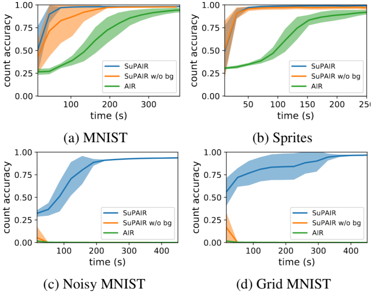
$$p_{\mathbf{m}}(\mathbf{q} \mid \mathbf{e}) = \frac{p_{\mathbf{m}}(\mathbf{q}, \mathbf{e})}{p_{\mathbf{m}}(\mathbf{e})}$$



© fineartamerica.com

Tractable MAR: scene understanding





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

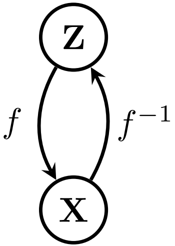
Stelzner et al., “Faster Attend-Infer-Repeat with Tractable Probabilistic Models”, 2019
Kossen et al., “Structured Object-Aware Physics Prediction for Video Modeling and Planning”, 2019

Normalizing flows

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

- an explicit likelihood!
- ...plus structured Jacobians

$\Longrightarrow$ *tractable EVI queries!*



~~Normalizing flows~~

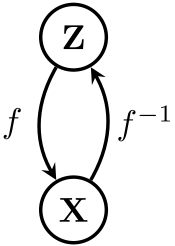
$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Z}}(f^{-1}(\mathbf{x})) \left| \det \left(\frac{\delta f^{-1}}{\delta \mathbf{x}} \right) \right|$$

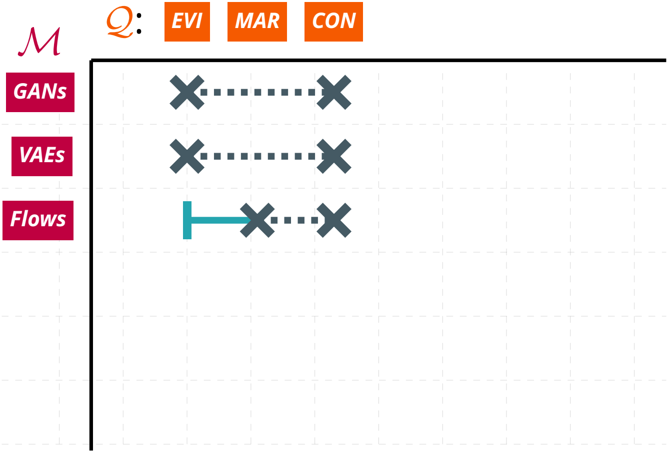
- an explicit likelihood!
- ...plus structured Jacobians

  $\Longrightarrow$ *tractable EVI queries!*

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

  $\Longrightarrow$ *unless f is "simple", e.g. bijection*



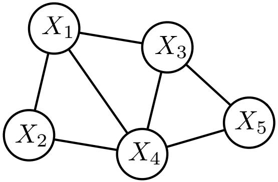


tractable bands

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-hard*

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

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

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

Why? Treewidth!

Treewidth:

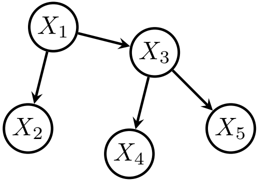
Informally, how tree-like is the graphical model $\mathbf{m}$?

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

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

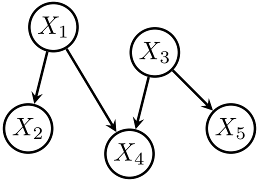
[Dechter 1998; Koller et al. 2009]. $\Longrightarrow$ *what about bounding the treewidth by design?*

Low-treewidth PGMs



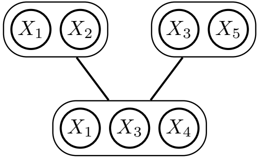
Trees

[Meilă et al. 2000]



Polytrees

[Dasgupta 1999]



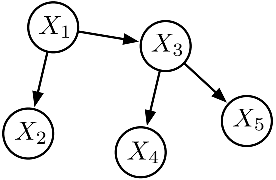
Thin Junction trees

[Bach et al. 2001]

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

Tree distributions

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

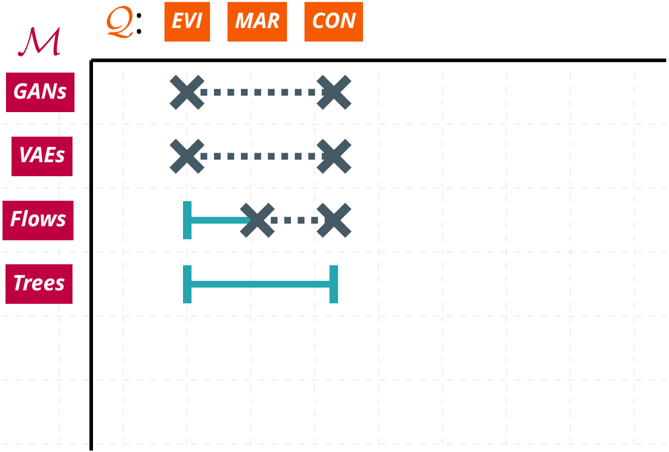


$$p(\mathbf{X}) = \prod_{i=1}^{n} p(x_i|\mathrm{Pa}_{x_i})$$

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

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

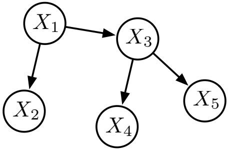
[1] *Chow et al., "Approximating discrete probability distributions with dependence trees", 1968*



tractable bands

What do we lose?

Expressiveness: Ability to represent rich and complex classes of distributions

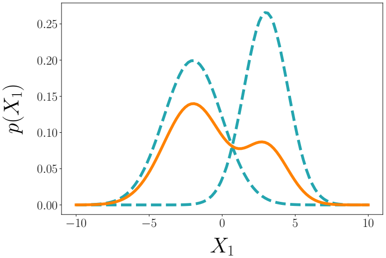


Bounded-treewidth PGMs lose the ability to represent *all possible distributions* ...

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

Mixtures

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

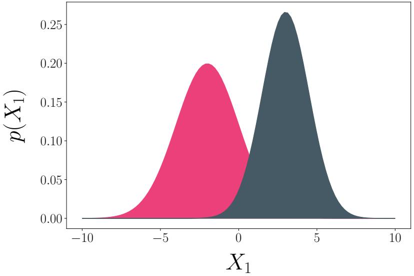


$$p(X) = w_1 \cdot p_1(X) + w_2 \cdot p_2(X)$$

EVI, MAR, CON queries scale linearly in k

Mixtures

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



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

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

$\Longrightarrow$ *increased expressiveness*

Expressiveness and efficiency

Expressiveness: Ability to represent rich and effective classes of functions

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

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

Expressiveness and efficiency

Expressiveness: Ability to represent rich and effective classes of functions

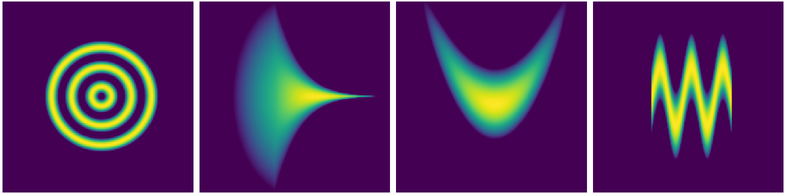
$\Longrightarrow$ *mixture of Gaussians can approximate any distribution!*

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

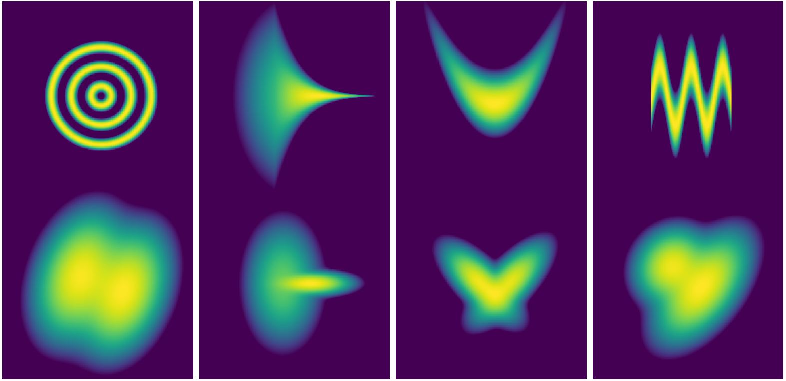
$\Longrightarrow$ *but how many components does a Gaussian mixture need?*

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

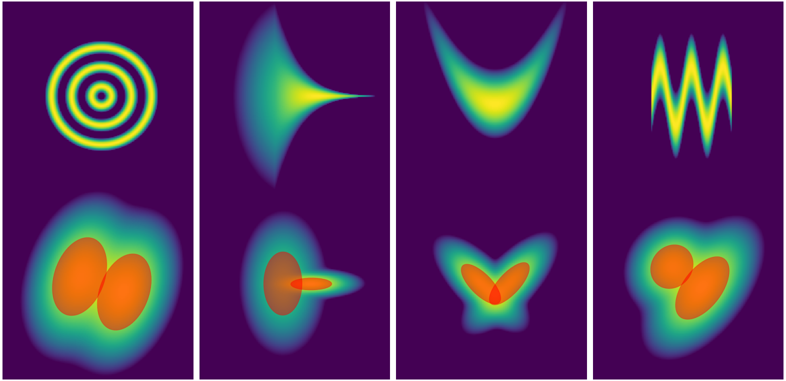
How expressive efficient are mixtures?



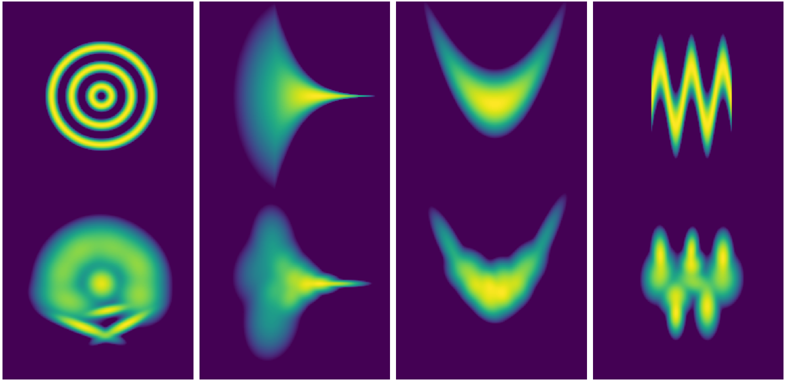
How expressive efficient are mixtures?



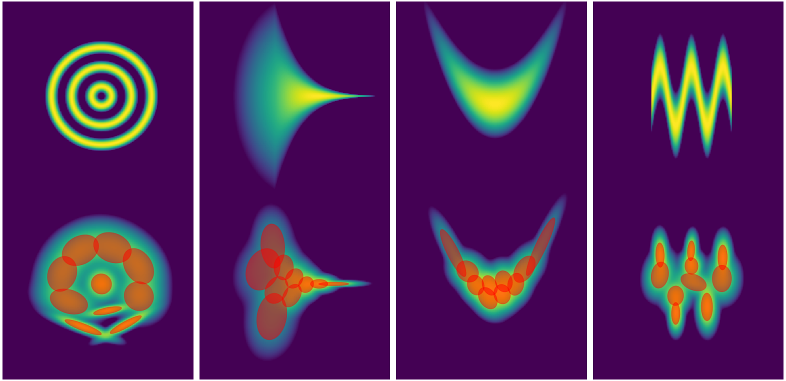
How expressive efficient are mixtures?



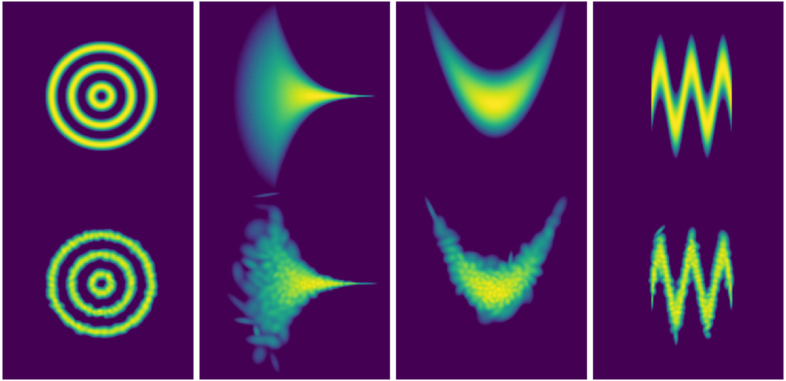
How expressive efficient are mixtures?



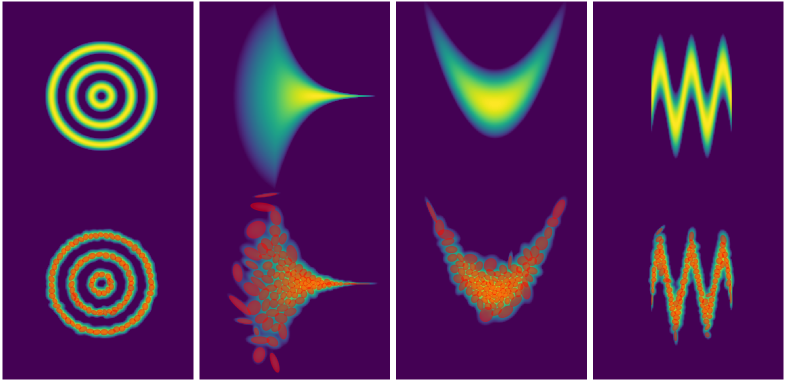
How expressive efficient are mixtures?



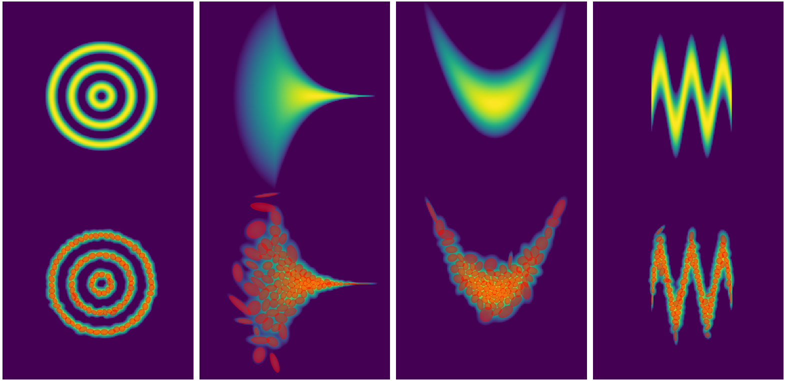
How expressive efficient are mixtures?



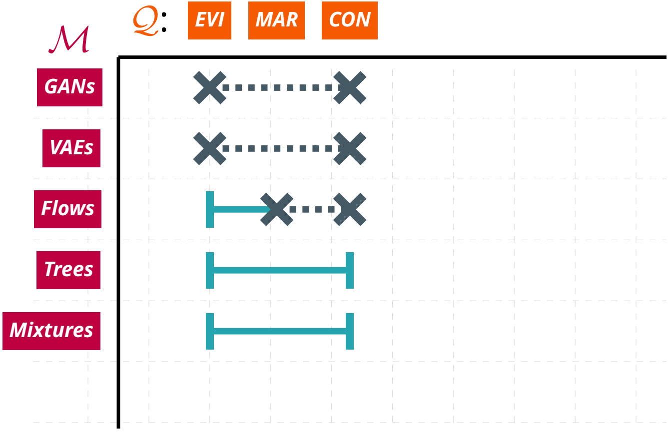
How expressive efficient are mixtures?



How expressive efficient are mixtures?



$\Rightarrow$ *stack mixtures like in deep generative models*

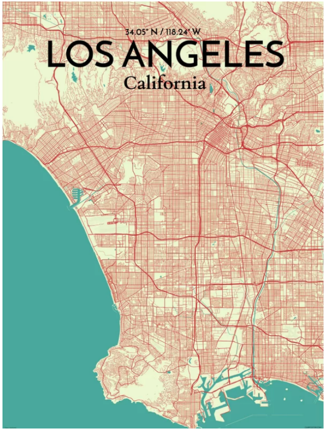


tractable bands

Maximum A Posteriori (MAP)

aka Most Probable Explanation (MPE)

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

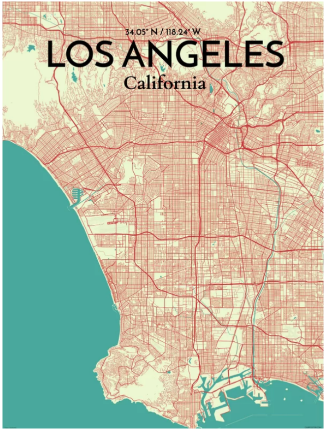


Maximum A Posteriori (MAP)

aka Most Probable Explanation (MPE)

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

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



© fineartamerica.com

Maximum A Posteriori (MAP)

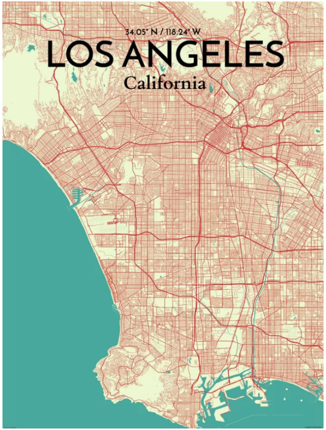
aka Most Probable Explanation (MPE)

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

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

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

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



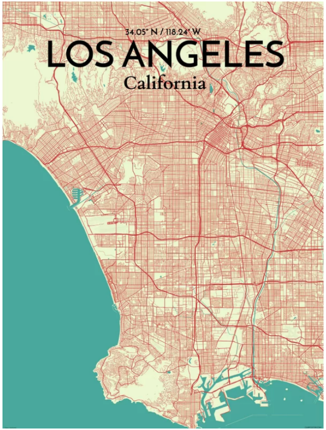
Maximum A Posteriori (MAP)

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!

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

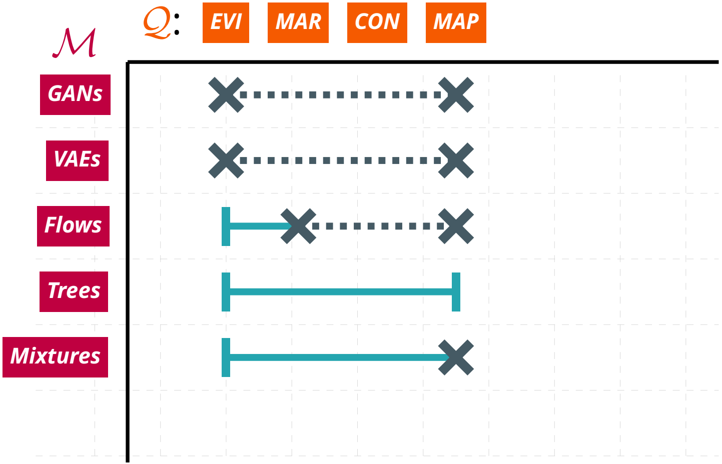


© fineartamerica.com

MAP inference : image inpainting

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

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

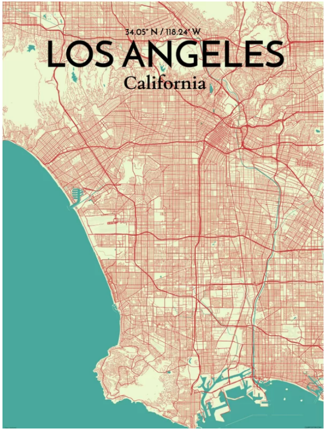


tractable bands

Marginal MAP (MMAP)

aka Bayesian Network MAP

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



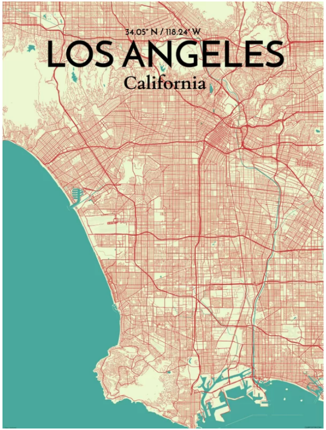
© fineartamerica.com

Marginal MAP (MMAP)

aka Bayesian Network MAP

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

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



***Marginal MAP* (MMAP)**

aka Bayesian Network MAP

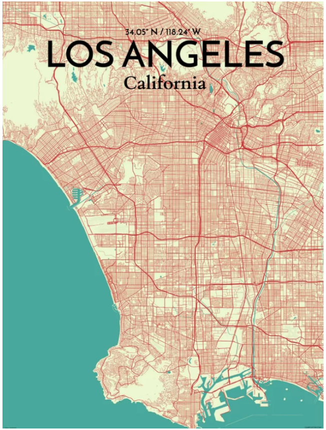
$\mathbf{q}_6$: *Which combination of roads is most likely to be jammed ~~on Monday~~ at 9am?*

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

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

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

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



© fineartamerica.com

Marginal MAP **(MMAP)**

aka Bayesian Network MAP

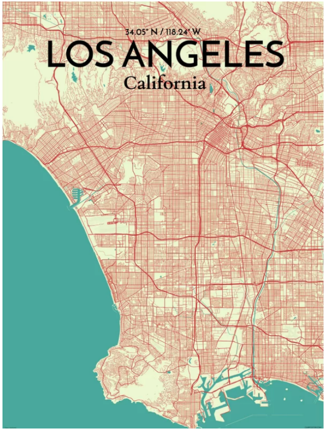
$\mathbf{q}_6$: *Which combination of roads is most likely to be jammed ~~on Monday~~ at 9am?*

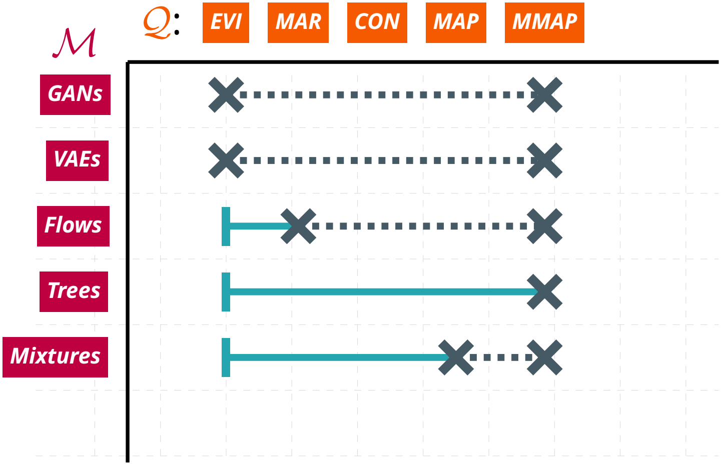
$$\mathbf{q}_6(\mathbf{m}) = \text{argmax}_{\mathbf{j}}\ p_{\mathbf{m}}(\mathbf{j}_1, \mathbf{j}_2, \ldots \mid \text{Time}=9)$$

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

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

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

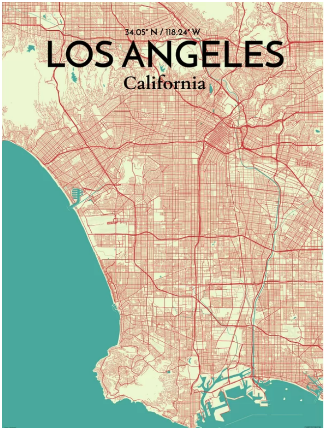




tractable bands

Advanced queries

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



© fineartamerica.com

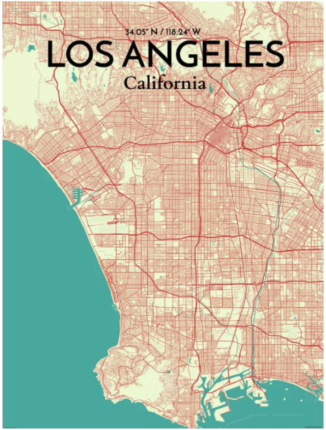
Bekker et al., "Tractable Learning for Complex Probability Queries", 2015

Advanced queries

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

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

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



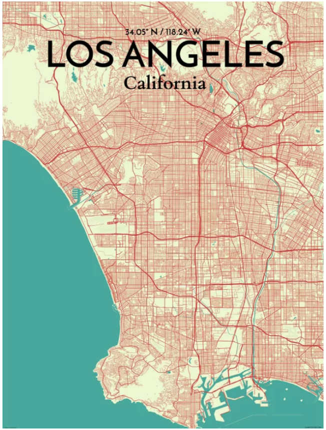
© fineartamerica.com

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

Advanced queries

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

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



© fineartamerica.com

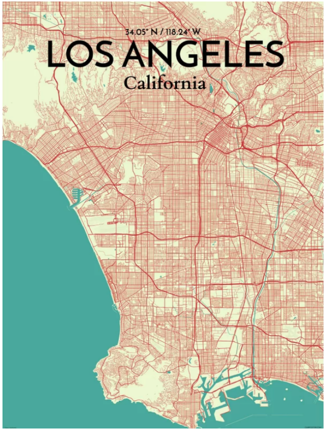
Bekker et al., "Tractable Learning for Complex Probability Queries", 2015

Advanced queries

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

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

$\Longrightarrow$ **counts + group comparison**



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

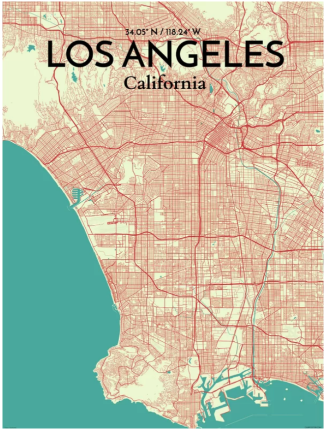
Advanced queries

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

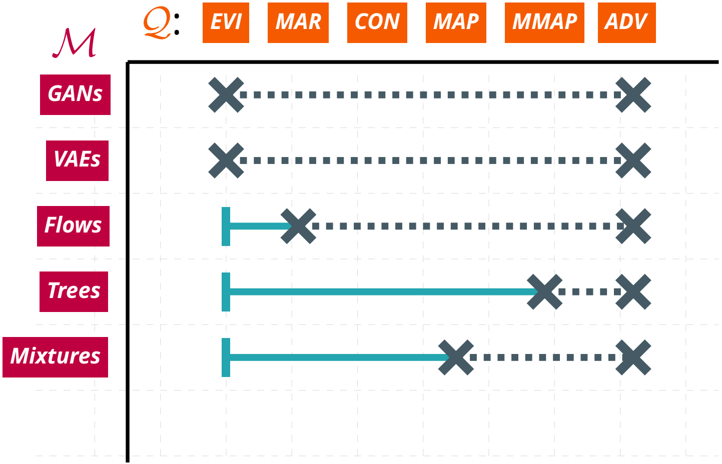
q_7: *What is the probability of seeing more traffic jams in Westwood than Hollywood?*

and more:

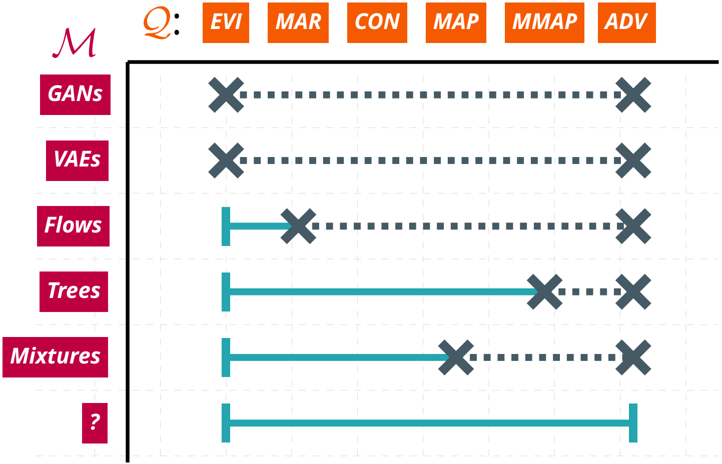
- expected classification agreement *[Oztok et al. 2016; Choi et al. 2017, 2018]*
- expected predictions *[Khosravi et al. 2019c]*



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



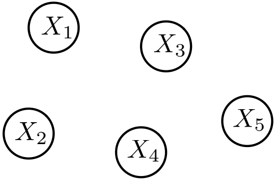
tractable bands



tractable bands

Fully factorized models

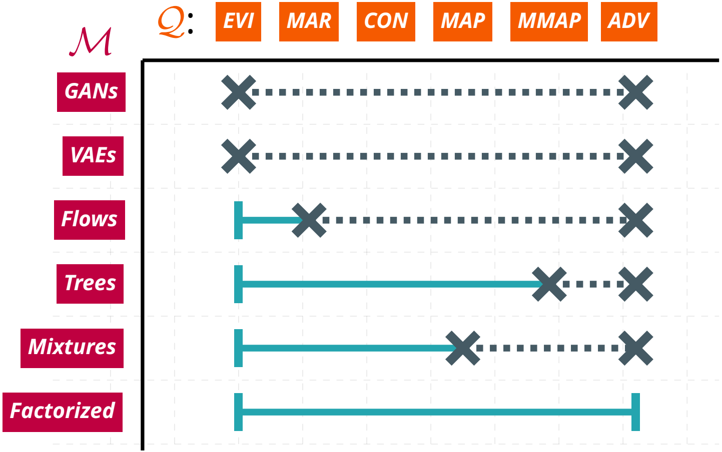
A completely disconnected graph. Example: Product of Bernoullis (PoBs)



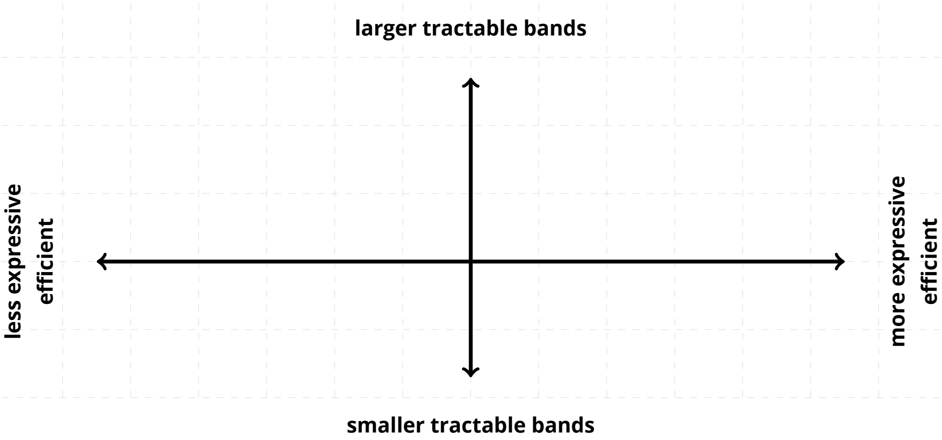
$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i)$$

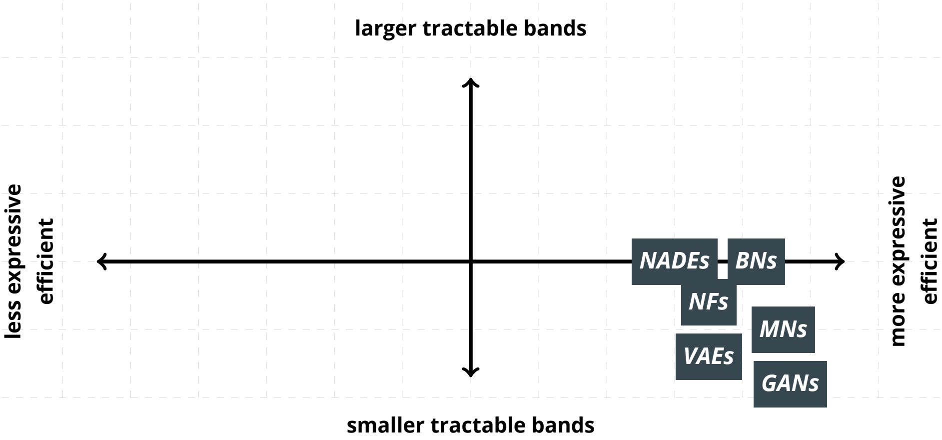
Complete evidence, marginals and MAP, MMAP inference is ***linear***!

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

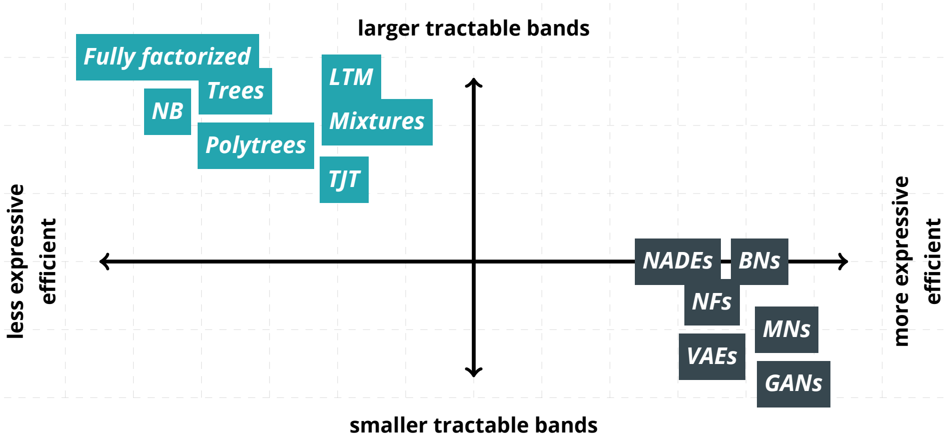


tractable bands

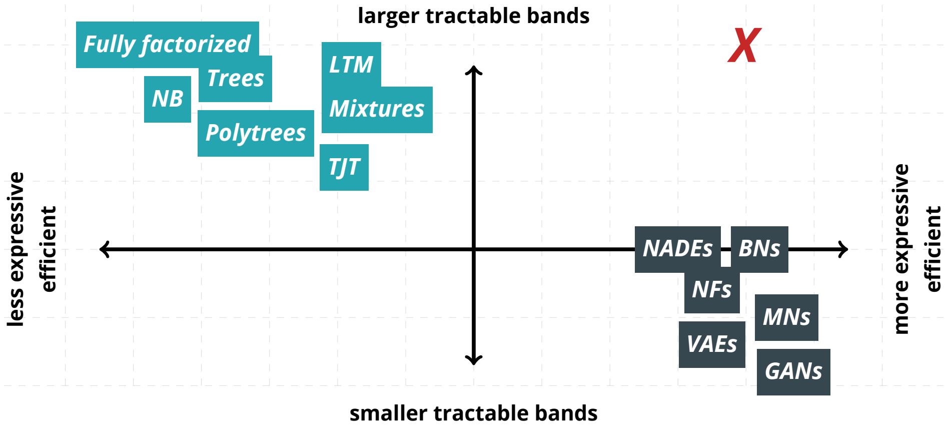

larger tractable bands
less expressive
efficient
more expressive
efficient
smaller tractable bands



Expressive models are not very tractable...



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



probabilistic circuits are at the "sweet spot"

Probabilistic Circuits

Probabilistic circuits

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

Probabilistic circuits

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

$\Longrightarrow$ operational semantics!

Probabilistic circuits

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

$\implies$ operational semantics!

$\implies$ by constraining the graph we can guarantee tractable inference...

Stay tuned for...

Next:

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

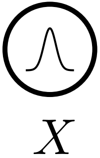
   ⇒ *How to build a tractable computational graph?*

2. For which queries are probabilistic circuits tractable?

   ⇒ *tractable classes induced by structural properties*

After: *How can probabilistic circuits be learned?*

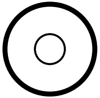
Distributions as computational graphs



Base case: a single node encoding a distribution

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

Distributions as computational graphs

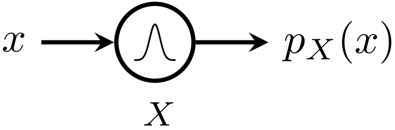


$\neg X$

Base case: a single node encoding a distribution

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

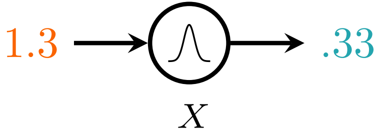
Distributions as computational graphs



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



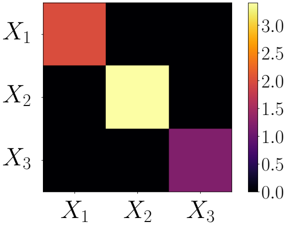
Simple distributions are tractable “black boxes” for:

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

Factorizations

Divide and conquer complexity

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

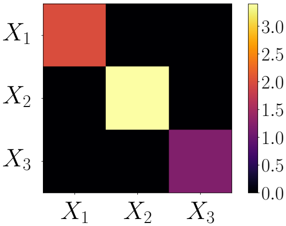


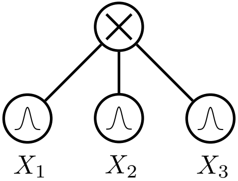
$\Rightarrow$ *e.g. modeling a multivariate Gaussian with diagonal covariance matrix...*

Factorizations are product nodes

Divide and conquer complexity

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



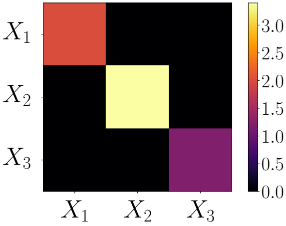


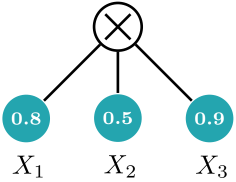
$\Longrightarrow$ *...with a product node over some univariate Gaussian distribution*

Factorizations are product nodes

Divide and conquer complexity

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



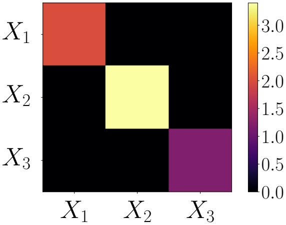


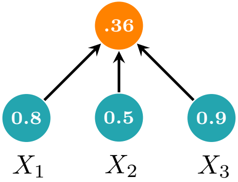
$\Longrightarrow$ *feedforward evaluation*

Factorizations are product nodes

Divide and conquer complexity

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

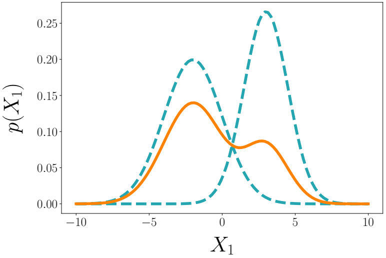




$\Longrightarrow$ *feedforward evaluation*

Mixtures

Enhance expressiveness

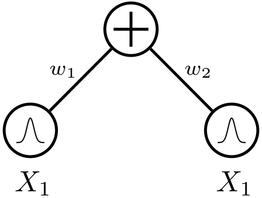


$$p(X) = w_1 \cdot p_1(X) + w_2 \cdot p_2(X)$$

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

Mixtures are sum nodes

Enhance expressiveness

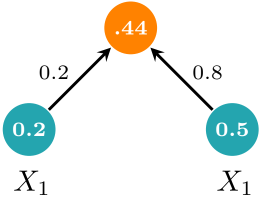


$$p(x) = 0.2 \cdot p_1(x) + 0.8 \cdot p_2(x)$$

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

Mixtures are sum nodes

Enhance expressiveness



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

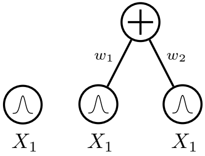
$\Longrightarrow$ *by* ***stacking*** *them we increase expressive efficiency*

A grammar for tractable models

Recursive semantics of probabilistic circuits

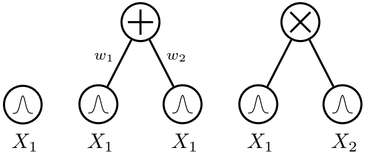
A grammar for tractable models

Recursive semantics of probabilistic circuits



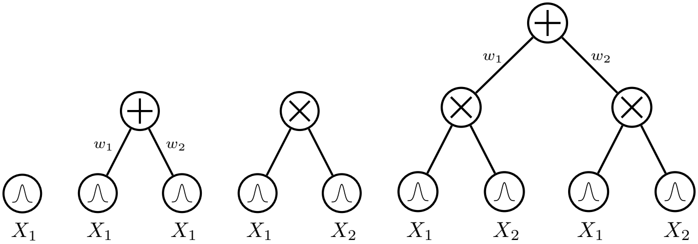
A grammar for tractable models

Recursive semantics of probabilistic circuits



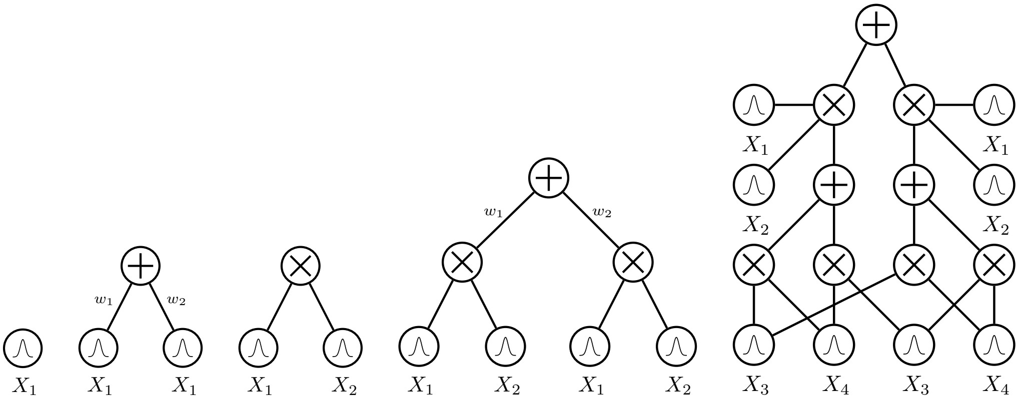
A grammar for tractable models

Recursive semantics of probabilistic circuits



A grammar for tractable models

Recursive semantics of probabilistic circuits



Building PCs in Python with SPFlow

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

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

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

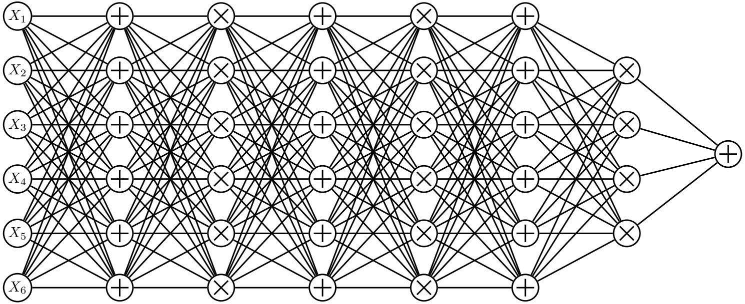
Probabilistic circuits are not PGMs!

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

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

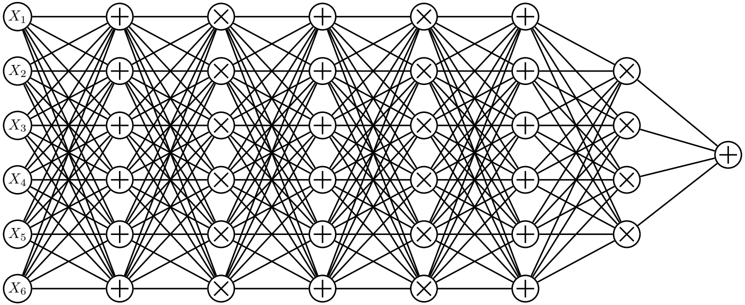
$\Rightarrow$ *they are **computational graphs**, more like neural networks*

Just sum, products and distributions?



just arbitrarily compose them like a neural network!

Just sum, products and distributions?



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

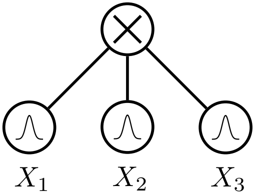
$\Longrightarrow$ structural constraints needed for tractability

Which structural constraints ensure tractability?

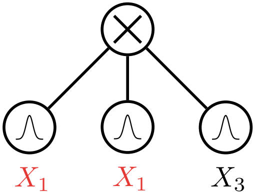
Decomposability

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

⇒ *just like in factorization!*



decomposable circuit



non-decomposable circuit

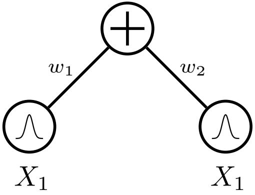
Darwiche and Marquis, "A knowledge compilation map", 2002

Smoothness

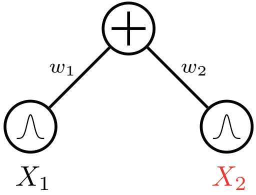
aka completeness

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

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



smooth circuit



non-smooth circuit

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

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

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

Computing arbitrary integrations (or summations)

$\Longrightarrow$ *linear in circuit size!*

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

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

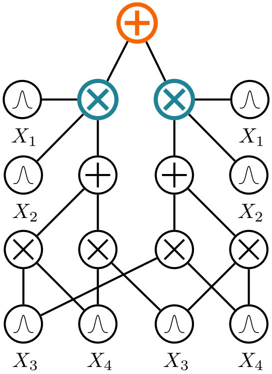
Smoothness + decomposability = tractable MAR

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

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

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

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



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

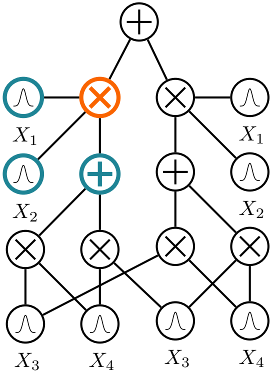
If $p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{x})p(\mathbf{y})p(\mathbf{z})$, (***decomposability***):

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

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

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

$\Longrightarrow$ *integrals decompose into easier ones*



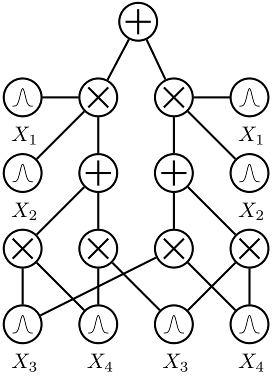
Smoothness + **decomposability** = **tractable MAR**

Forward pass evaluation for MAR

$\Longrightarrow$ *linear in circuit size!*

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

- leafs over X_1 and X_3 output $Z_i = \int p(x_i)dx_i$
  - $\Longrightarrow$ *for normalized leaf distributions:* 1.0
- leafs over X_2 and X_4 output **EVI**
- feedforward evaluation (bottom-up)



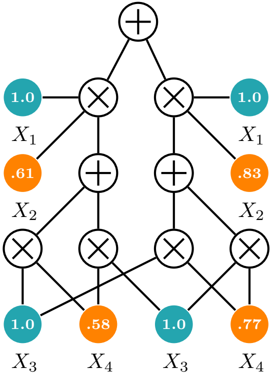
Smoothness + decomposability = tractable MAR

Forward pass evaluation for MAR

$\Longrightarrow$ *linear in circuit size!*

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

- leafs over X_1 and X_3 output $Z_i = \int p(x_i)dx_i$
  - $\Longrightarrow$ *for normalized leaf distributions:* 1.0
- leafs over X_2 and X_4 output ***EVI***
- feedforward evaluation (bottom-up)



Smoothness + decomposability = tractable MAR

Forward pass evaluation for MAR

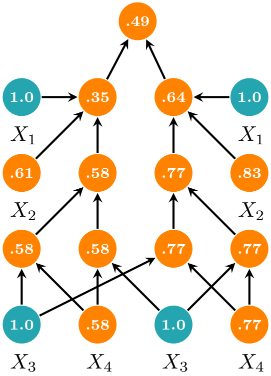
$\Rightarrow$ *linear in circuit size!*

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

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

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

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



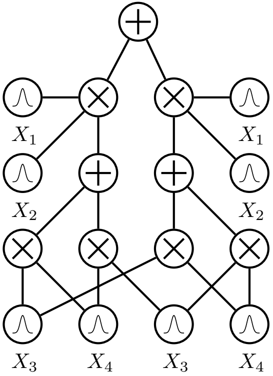
Smoothness + decomposability = tractable CON

Analogously, for arbitrary conditional queries:

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

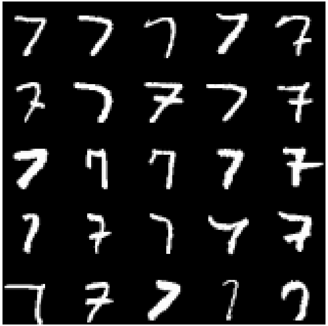
1. evaluate $p(\mathbf{q}, \mathbf{e})$ $\Longrightarrow$ *one feedforward pass*
2. evaluate $p(\mathbf{e})$ $\Longrightarrow$ *another feedforward pass*

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

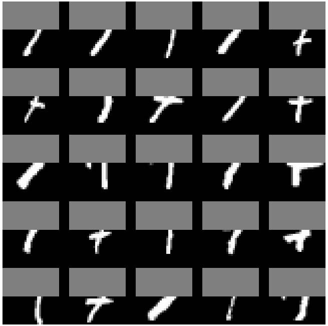


Tractable MAR on PCs (Einsum Networks)

EVI $10,958.72\,\text{nats}$



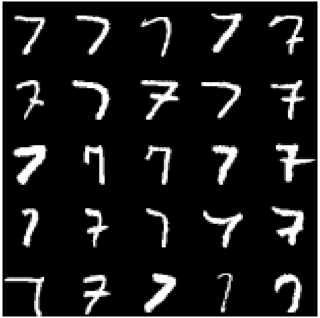
MAR $5,387.55\,\text{nats}$



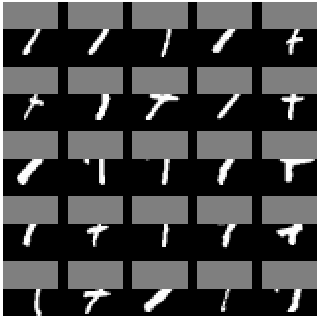
Peharz et al., "Einsum Networks: Fast and Scalable Learning of Tractable Probabilistic Circuits", 2020

Tractable CON on PCs (Einsum Networks)

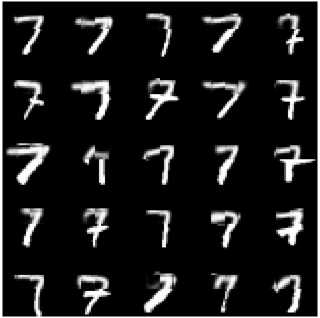
Original



Missing

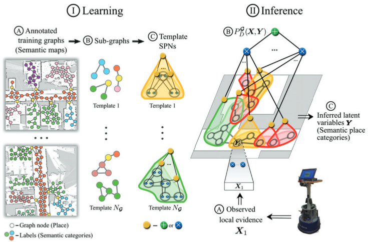


Conditional sample



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

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 and Rao, "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:

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

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

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

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

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

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

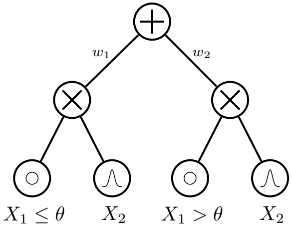
$\Longrightarrow$ *MAP for latent variable models is* **intractable** *[Conaty et al. 2017]*

Determinism

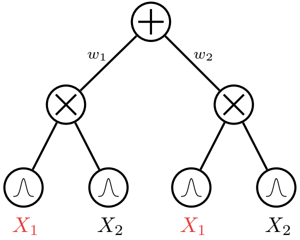
aka selectivity

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

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



deterministic circuit



non-deterministic circuit

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

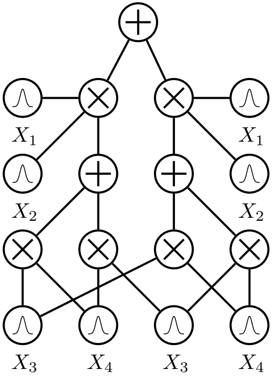
Determinism + decomposability = tractable MAP

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

$\Longrightarrow$ *linear in circuit size!*

E.g., suppose we want to compute:

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

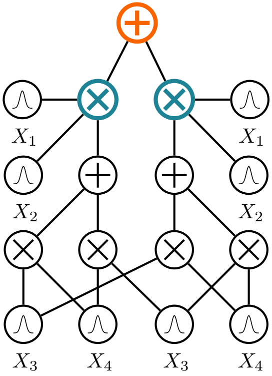


Determinism + **decomposability** = **tractable MAP**

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

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

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

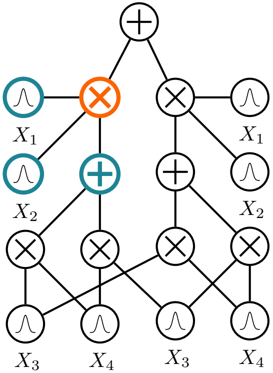


Determinism + **decomposability** = **tractable MAP**

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

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

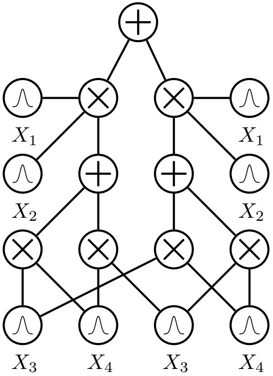
$\Longrightarrow$ *solving optimization independently*



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

Evaluating the circuit twice:

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

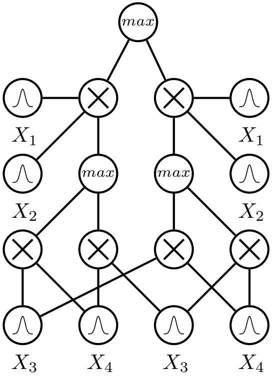


Determinism + **decomposability** = **tractable MAP**

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

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

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

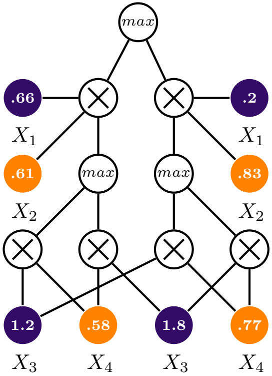


Determinism + ***decomposability*** = ***tractable MAP***

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

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

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

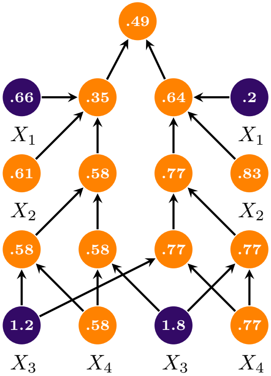


Determinism + **decomposability** = **tractable MAP**

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

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

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

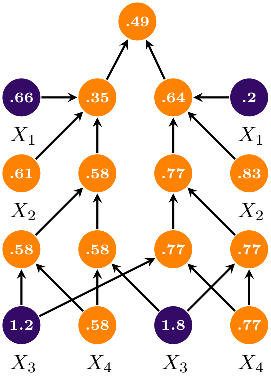


Determinism + **decomposability** = **tractable MAP**

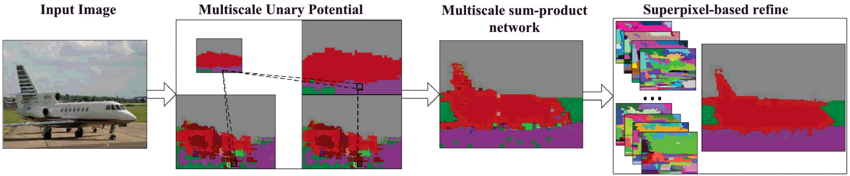
Evaluating the circuit twice:
bottom-up and ***top-down*** $\Rightarrow$ *still linear in circuit size!*

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

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



MAP inference : image segmentation



Semantic segmentation is MAP over joint pixel and label space

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

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

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

Friesen and Domingos, "Submodular Sum-product Networks for Scene Understanding", 2016

Determinism + decomposability = tractable MMAP

Analogously, we could also do a MMAP query?:

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

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

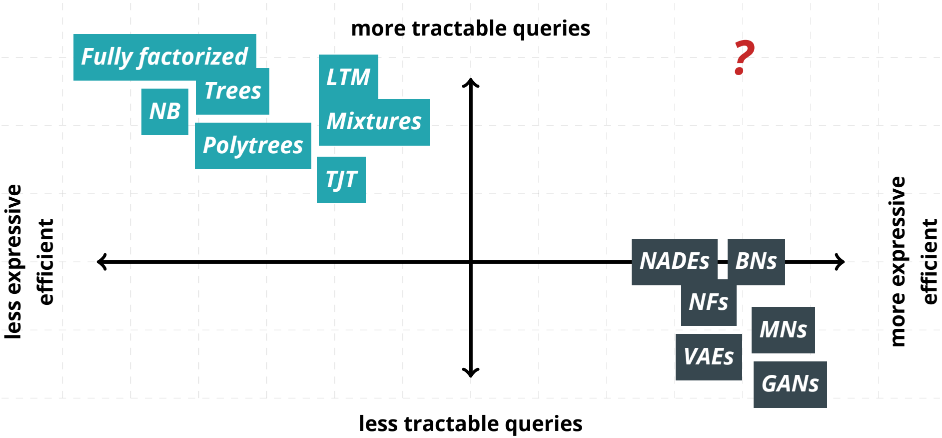
We ***cannot*** decompose a MMAP query!

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

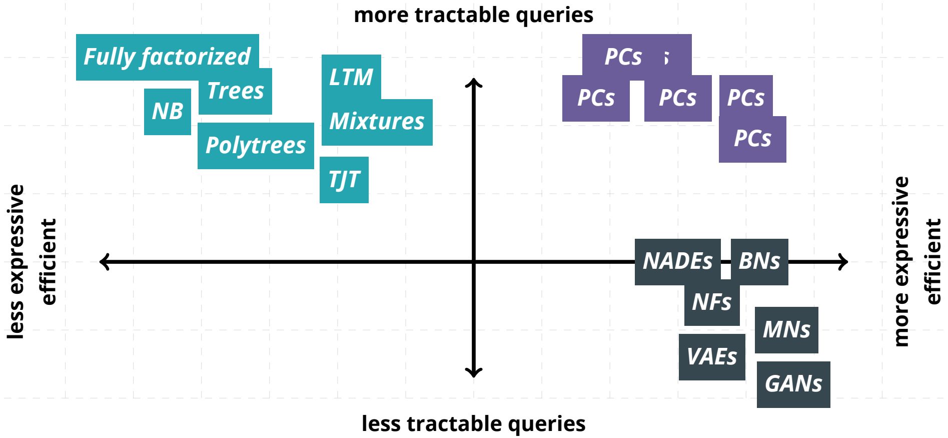
we still have latent variables to marginalize…

We need more structural properties!

$\Rightarrow$ *more advanced queries in Part 4 later…*



where are probabilistic circuits?



tractability vs expressive efficiency

Low-treewidh PGMs

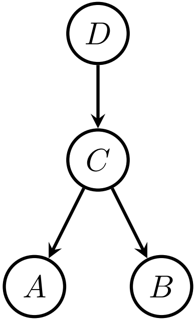
Tree, polytrees and Thin Junction trees can be turned into

- decomposable
- smooth
- deterministic

circuits

Therefore they support tractable

- EVI
- MAR/CON
- MAP



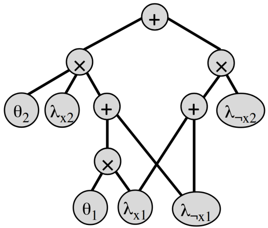
Arithmetic Circuits (ACs)

ACs *[Darwiche 2003]* are

- decomposable
- smooth
- deterministic

They support tractable

- EVI
- MAR/CON
- MAP



⇒ *parameters are attached to the leaves*

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

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

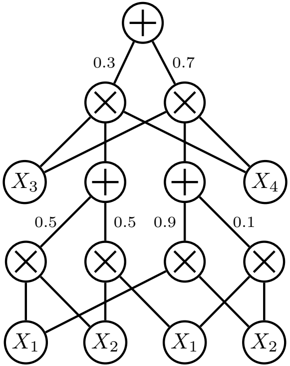
Sum-Product Networks (SPNs)

SPNs *[Poon et al. 2011]* are

- decomposable
- smooth
- ~~deterministic~~

They support tractable

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



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

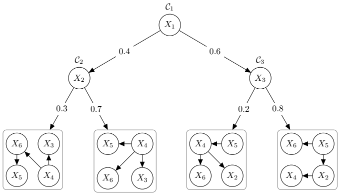
Cutset Networks (CNets)

CNets *[Rahman et al. 2014]* are

- decomposable
- smooth
- deterministic

They support tractable

- EVI
- MAR/CON
- MAP



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

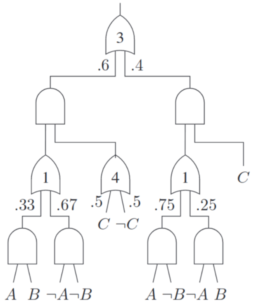
Probabilistic Sentential Decision Diagrams

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

- structured decomposable
- smooth
- deterministic

They support tractable

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



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

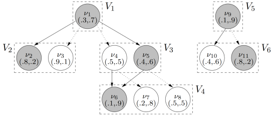
Probabilistic Decision Graphs

PDGs *[Jaeger 2004]* are

- structured decomposable
- smooth
- deterministic

They support tractable

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



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

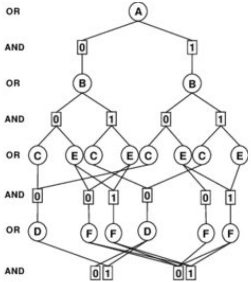
AndOrGraphs

AndOrGarphs *[Dechter et al. 2007]* are

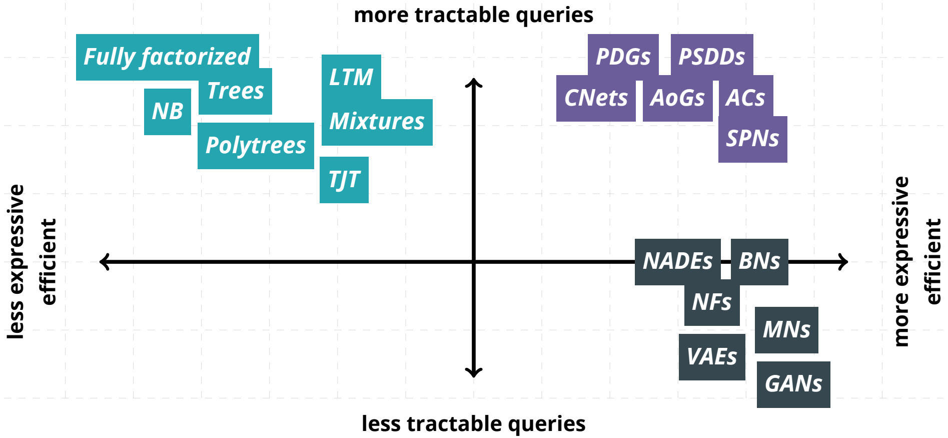
- structured decomposable
- smooth
- deterministic

They support tractable

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



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



tractability vs expressive efficiency

Are all tractable models PCs?

- In some sense: *Yes, they can always be leaf distributions!*
- More interesting: Can all tractable probabilistic models be written as compact PCs with structural constraints over "simple" leafs?
- Concretely: Can all binary distributions that are tractable for MAR be written as smooth decomposable PCs with univariate leafs?

Are all tractable models PCs?

- In some sense: *Yes, they can always be leaf distributions!*
- More interesting: Can all tractable probabilistic models be written as compact PCs with structural constraints over "simple" leafs?
- Concretely: Can all binary distributions that are tractable for MAR be written as smooth decomposable PCs with univariate leafs?

Are all tractable models PCs?

- In some sense: *Yes, they can always be leaf distributions!*
- More interesting: Can all tractable probabilistic models be written as compact PCs with structural constraints over "simple" leafs?
- Concretely: Can all binary distributions that are tractable for MAR be written as smooth decomposable PCs with univariate leafs?

Are all tractable models PCs?

- In some sense: *Yes, they can always be leaf distributions!*
- More interesting: Can all tractable probabilistic models be written as compact PCs with structural constraints over "simple" leafs?
- Concretely: Can all binary distributions that are tractable for MAR be written as smooth decomposable PCs with univariate leafs?

Almost! One possible exception are Determinantal Point Proceses (DPPs).
Active area of research *[Martens et al. 2014; Zhang et al. 2020]*

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 and Domingos, "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

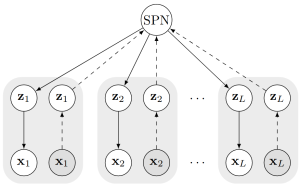
How expressive are probabilistic circuits?

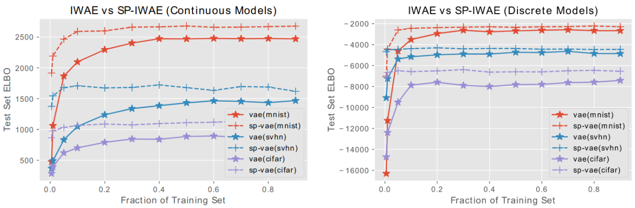
density estimation benchmarks

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

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*

Tan and Peharz, "Hierarchical Decompositional Mixtures of Variational Autoencoders", 2019

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* $\mathbf{\Omega}$

Learning probabilistic circuits

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

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

Learning probabilistic circuits

	Parameters	Structure
Generative	?	?
Discriminative	?	?

Stay tuned for...

Next:

1. *How to learn circuit parameters?*

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

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

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

After: *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

X_1

The simplest PC: a single input distribution p_{L} with parameters $\boldsymbol{\theta}$

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

Learning input distributions

As simple as tossing a coin

X_1

The simplest PC: a single input distribution p_{L} with parameters $\boldsymbol{\theta}$

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

E.g. Bernoulli with parameter θ

$$\hat{\theta}_{\mathsf{ML}} = \frac{\sum_{x \in \mathcal{D}} \mathbb{1}[x = 1] + \alpha}{|\mathcal{D}| + 2\alpha} \quad \Longrightarrow \quad \textit{Laplace smoothing}$$

Learning input distributions

General case: still simple

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

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

Learning input distributions

General case: still simple

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

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

Where:

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

Learning input distributions

General case: still simple

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

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

Where:

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

Learning input distributions

General case: still simple

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

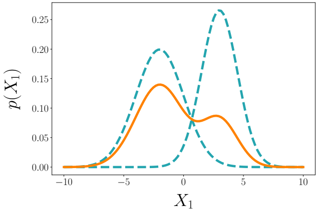
$$p_{\mathsf{L}}(\mathbf{x}) = \boldsymbol{h}(\mathbf{x}) \; \exp(\boldsymbol{T}(\mathbf{x})^T \boldsymbol{\theta} - A(\boldsymbol{\theta}))$$

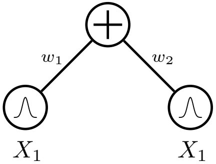
Maximum likelihood estimation is still "***counting***":

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

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

The simplest "real" PC: a sum node

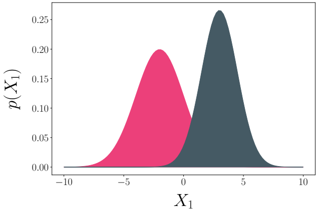


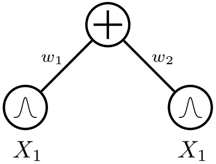


Recall that sum nodes represent ***mixture models***:

$$p_{\mathsf{S}}(\mathbf{x}) = \sum_{k=1}^{K} w_k p_{\mathsf{L}_k}(\mathbf{x})$$

The simplest "real" PC: a sum node





Recall that sum nodes represent ***latent variable models***:

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

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.

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

Expectation-Maximization for mixtures

- $\boldsymbol{\theta}^{new} \leftarrow \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{p(Z \mid \mathbf{X}; \boldsymbol{\theta}^{old})} [\log p(\mathbf{X}, Z; \boldsymbol{\theta})]$
- ML if Z was observed:

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

- Z is unobserved—but we have $p(Z = k \mid \mathbf{x}) \propto w_k \, \mathsf{L}_k(\mathbf{x})$.

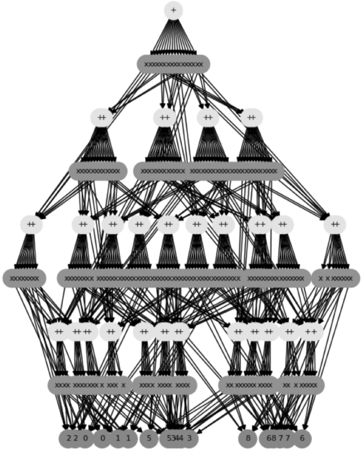
$$w_k^{new} = \frac{\sum_{\mathbf{x} \in \mathcal{D}} p(Z = k \mid \mathbf{x})}{|\mathcal{D}|} \qquad \phi_k^{new} = \frac{\sum_{\mathbf{x}, z \in \mathcal{D}} p(Z = k \mid \mathbf{x}) T(\mathbf{x})}{\sum_{z \in \mathcal{D}} p(Z = k \mid \mathbf{x})}$$

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 ...

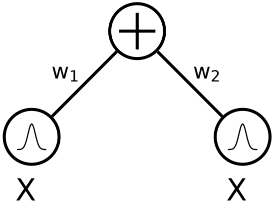
Expectation-Maximization for PCs

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



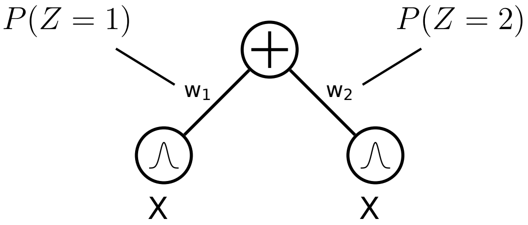
Expectation-Maximization for PCs

[Peharz et al. 2016]



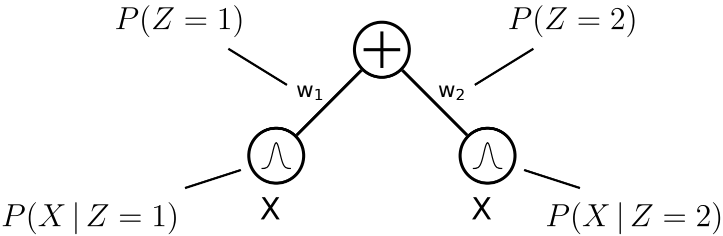
Expectation-Maximization for PCs

[Peharz et al. 2016]



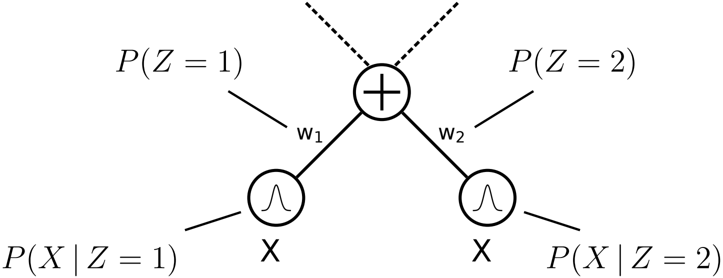
Expectation-Maximization for PCs

[Peharz et al. 2016]



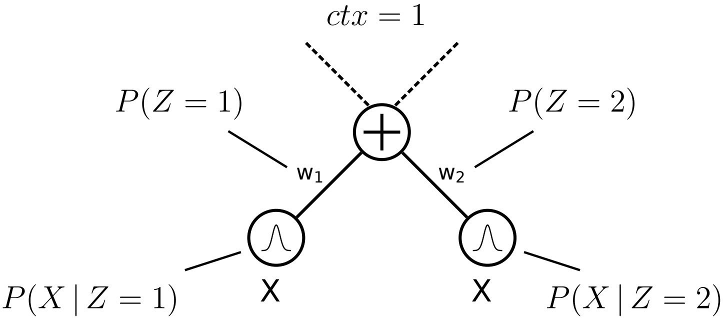
Expectation-Maximization for PCs

[Peharz et al. 2016]



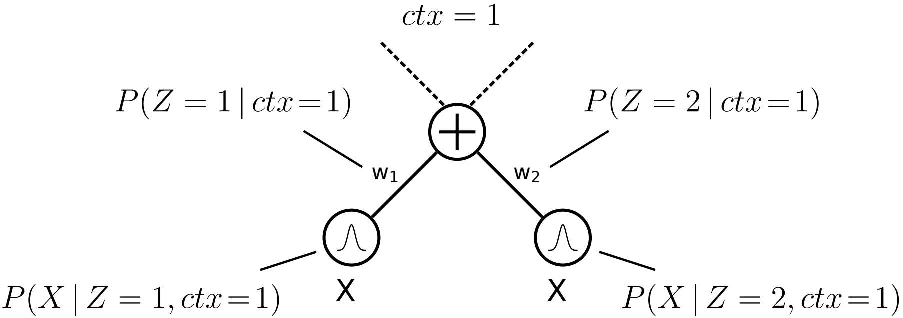
Expectation-Maximization for PCs

[Peharz et al. 2016]



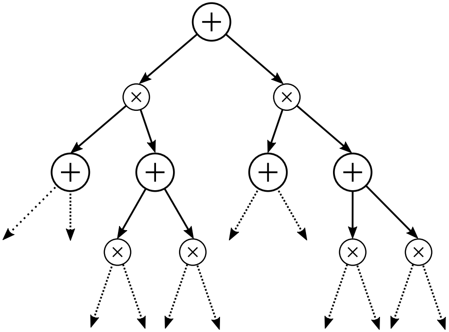
Expectation-Maximization for PCs

[Peharz et al. 2016]



Expectation-Maximization

Tractable MAR (smooth, decomposable)

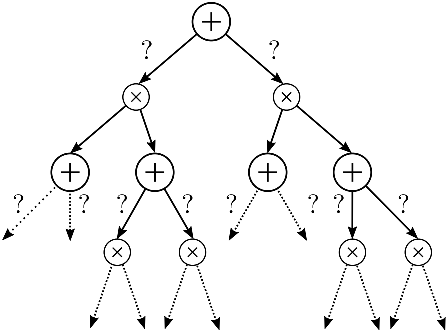


For learning, we need to know for each sum S:

1. Is S reached ($ctx = ?$)
2. Which child does it select ($Z_{\mathsf{S}} = ?$)

Expectation-Maximization

Tractable MAR (smooth, decomposable)

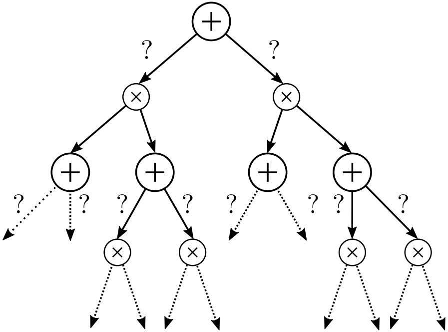


For learning, we need to know for each sum S:

1. Is S reached ($ctx = ?$)
2. Which child does it select ($Z_{\mathsf{S}} = ?$)

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 ($Z_\mathsf{S} = ?$)

We can ***infer*** it: $p(ctx, Z_\mathsf{S} \mid \mathbf{x})$

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)

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

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

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

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)

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

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

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

$\Longrightarrow$ *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:

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

$\Longrightarrow$ *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)

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

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

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

$\Longrightarrow$ *also derivable from a concave-convex procedure (CCCP)* *[Zhao et al. 2016b]*

Darwiche, "A Differential Approach to Inference in Bayesian Networks", 2003
Peharz et al., "On the Latent Variable Interpretation in Sum-Product Networks", 2016

EM with Einsum Networks @PyTorch

Creating a PC as an EinsumNetwork *[Peharz et al. 2020]* for MNIST

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

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

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

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

EM with Einsum Networks @PyTorch

...and training its parameters with EM

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

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

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

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

EM with Einsum Networks @PyTorch

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

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

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

Expectation-Maximization

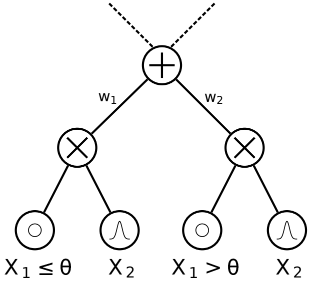
Tractable MAR/MAP (smooth, decomposable, deterministic)

~~Expectation-Maximization~~ Exact ML

Tractable MAR/MAP (smooth, decomposable, deterministic)

Exact ML

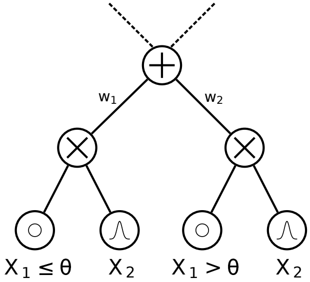
Tractable MAR/MAP (smooth, decomposable, deterministic)



Exact ML

Tractable MAR/MAP (smooth, decomposable, deterministic)

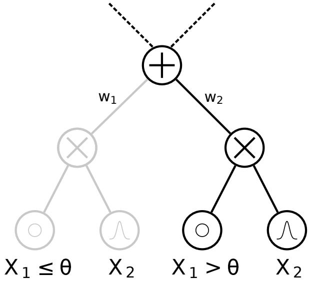
Deterministic circuit $\Rightarrow$ at most one non-zero sum child (for complete input).



Exact ML

Tractable MAR/MAP (smooth, decomposable, deterministic)

For example, the second child of this sum node...

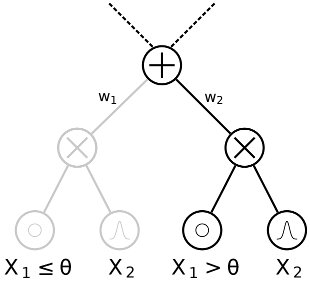


Exact ML

Tractable MAR/MAP (smooth, decomposable, deterministic)

For example, the second child of this sum node...

...but that rules out $Z = 1$! $\quad \Rightarrow P(Z = 2 \mid \mathbf{x}) = 1$

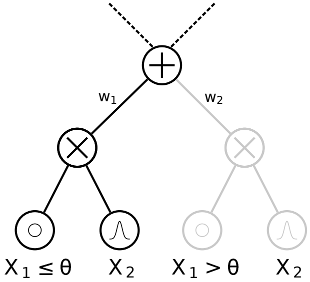


Exact ML

Tractable MAR/MAP (smooth, decomposable, deterministic)

Likewise, if the first child is non-zero:

$$\Rightarrow P(Z = 1 \,|\, \mathbf{x}) = 1$$



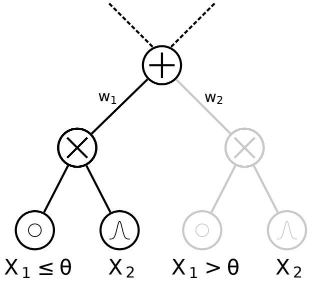
Exact ML

Tractable MAR/MAP (smooth, decomposable, deterministic)

Likewise, if the first child is non-zero:

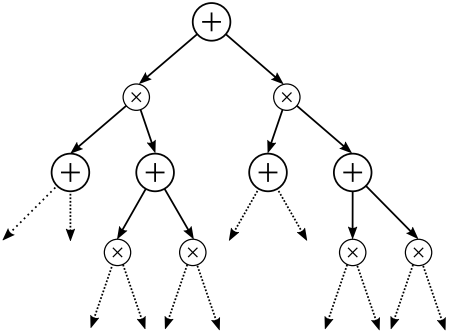
$\Rightarrow P(Z = 1 \,|\, \mathbf{x}) = 1$

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



Example

Tractable MAR/MAP (smooth, decomposable, deterministic)

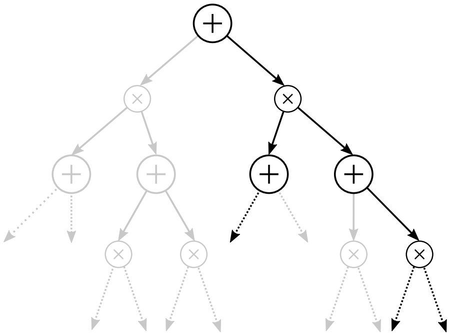


For each sum node, we know

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

Example

Tractable MAR/MAP (smooth, decomposable, deterministic)

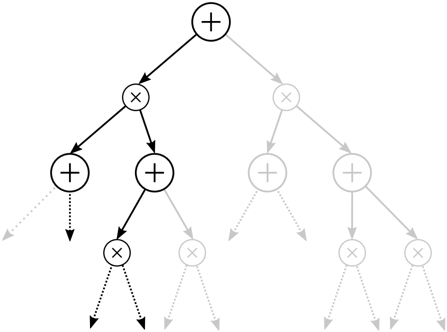


For each sum node, we know

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

Example

Tractable MAR/MAP (smooth, decomposable, deterministic)

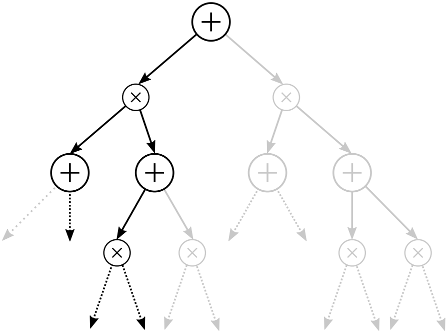


For each sum node, we know

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

$\Longrightarrow$ ***MLE** by **counting**!*

Exact ML

Tractable MAR/MAP (smooth, decomposable, deterministic)

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

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

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

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

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

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

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

$$w_{i,j}^{\text{ML}} = \frac{\sum_{\mathbf{x}\in\mathcal{D}} \mathbb{1}\{\mathbf{x} \models [i \wedge j]\}}{\sum_{\mathbf{x}\in\mathcal{D}} \mathbb{1}\{\mathbf{x} \models [i]\}} \qquad \begin{matrix} \leftarrow ctx_i = 1, Z_i = j \\ \leftarrow ctx_i = 1 \end{matrix}$$

$\Rightarrow$ *global maximum with single pass over* $\mathcal{D}$

$\Rightarrow$ *regularization, e.g. Laplace-smoothing, to avoid division by zero*

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

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

Training PCs in Julia with Juice.jl

Training maximum likelihood parameters of probabilistic circuits with determinism is incredibly fast.

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

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

---

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

Bayesian parameter learning

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

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

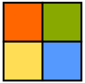
- Moment matching (oBMM) *[Jaini et al. 2016; Rashwan et al. 2016]*
- Collapsed variational inference algorithm *[Zhao et al. 2016a]*
- Gibbs sampling *[Trapp et al. 2019; Vergari et al. 2019]*

Learning probabilistic circuits

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

Image-tailored (handcrafted) structures

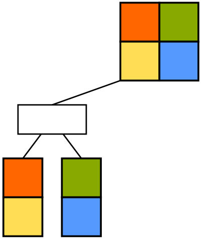
"Recursive Image Slicing"



Poon and Domingos, "Sum-Product Networks: a New Deep Architecture", 2011

Image-tailored (handcrafted) structures

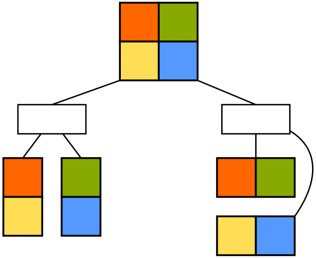
"Recursive Image Slicing"



Poon and Domingos, "Sum-Product Networks: a New Deep Architecture", 2011

Image-tailored (handcrafted) structures

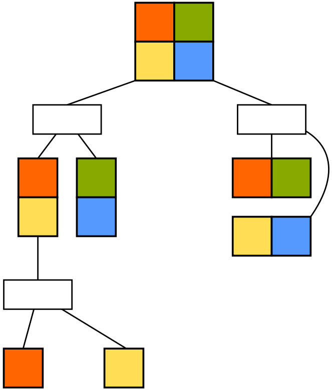
"Recursive Image Slicing"



Poon and Domingos, "Sum-Product Networks: a New Deep Architecture", 2011

Image-tailored (handcrafted) structures

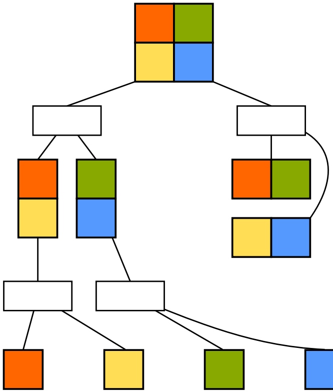
"Recursive Image Slicing"



Poon and Domingos, "Sum-Product Networks: a New Deep Architecture", 2011

Image-tailored (handcrafted) structures

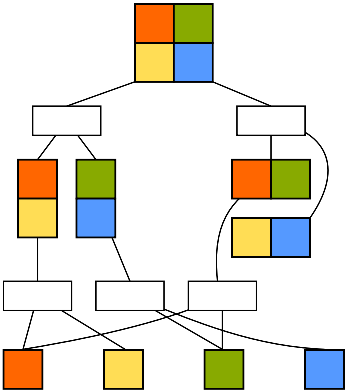
"Recursive Image Slicing"



Poon and Domingos, "Sum-Product Networks: a New Deep Architecture", 2011

Image-tailored (handcrafted) structures

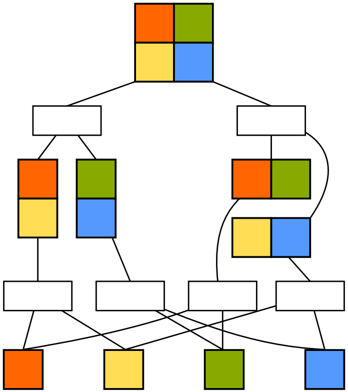
"Recursive Image Slicing"



Poon and Domingos, "Sum-Product Networks: a New Deep Architecture", 2011

Image-tailored (handcrafted) structures

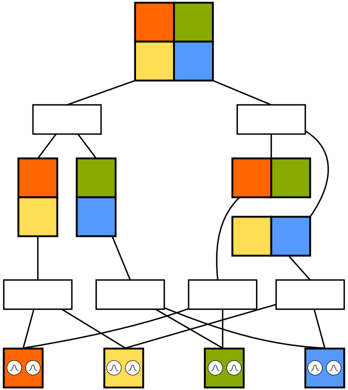
"Recursive Image Slicing"



Poon and Domingos, "Sum-Product Networks: a New Deep Architecture", 2011

Image-tailored (handcrafted) structures

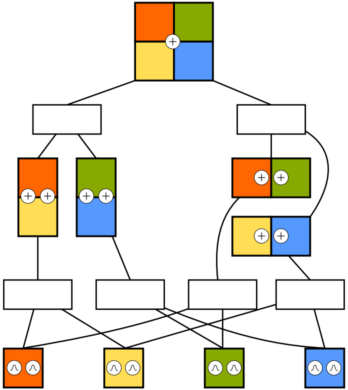
"Recursive Image Slicing"



Poon and Domingos, "Sum-Product Networks: a New Deep Architecture", 2011

Image-tailored (handcrafted) structures

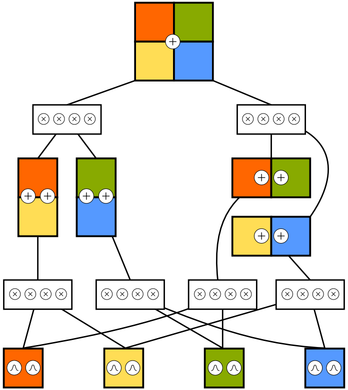
"Recursive Image Slicing"



Poon and Domingos, "Sum-Product Networks: a New Deep Architecture", 2011

Image-tailored (handcrafted) structures

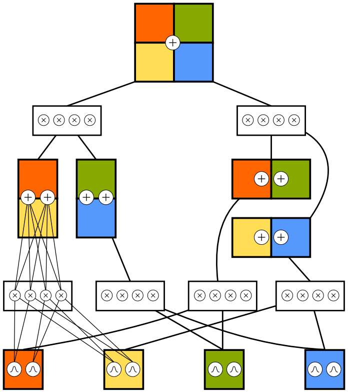
"Recursive Image Slicing"



Poon and Domingos, "Sum-Product Networks: a New Deep Architecture", 2011

Image-tailored (handcrafted) structures

"Recursive Image Slicing"

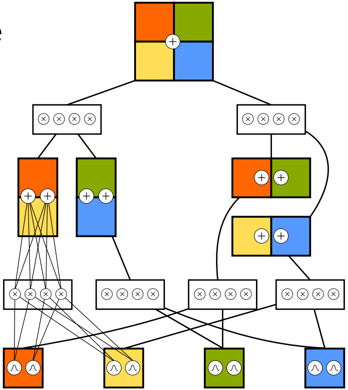


Poon and Domingos, "Sum-Product Networks: a New Deep Architecture", 2011

Image-tailored (handcrafted) structures

"Recursive Image Slicing"

⇒ Smooth & Decomposable



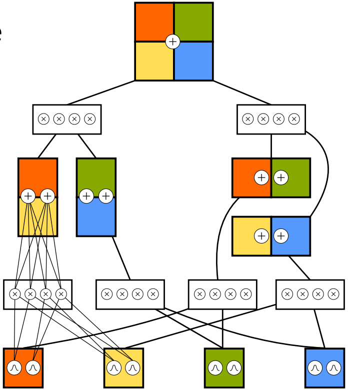
Poon and Domingos, "Sum-Product Networks: a New Deep Architecture", 2011

Image-tailored (handcrafted) structures

"Recursive Image Slicing"

⇒ Smooth & Decomposable

⇒ Tractable MAR

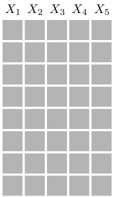


Poon and Domingos, "Sum-Product Networks: a New Deep Architecture", 2011

Learning the structure from data

"Recursive Data Slicing" — LearnSPN

Cluster

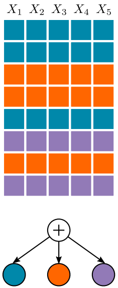


Gens and Domingos, "Learning the Structure of Sum-Product Networks", 2013

Learning the structure from data

"Recursive Data Slicing" — LearnSPN

Cluster $\longrightarrow$ ***sum node***

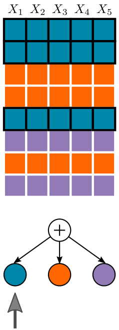


Gens and Domingos, "Learning the Structure of Sum-Product Networks", 2013

Learning the structure from data

"Recursive Data Slicing" — LearnSPN

Try to find independent groups of random variables



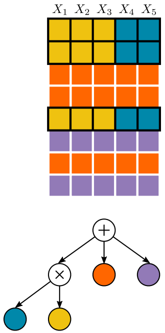
Gens and Domingos, "Learning the Structure of Sum-Product Networks", 2013

Learning the structure from data

"Recursive Data Slicing" — LearnSPN

Try to find independent groups of random variables

Success $\longrightarrow$ ***product node***

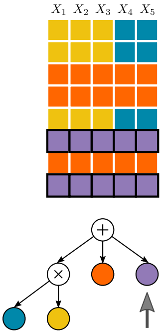


Gens and Domingos, "Learning the Structure of Sum-Product Networks", 2013

Learning the structure from data

“Recursive Data Slicing” — LearnSPN

Try to find independent groups of random variables

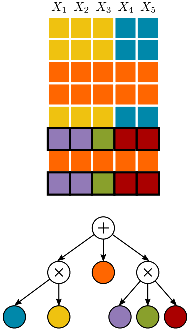


Learning the structure from data

"Recursive Data Slicing" — LearnSPN

Try to find independent groups of random variables

Success $\longrightarrow$ ***product node***

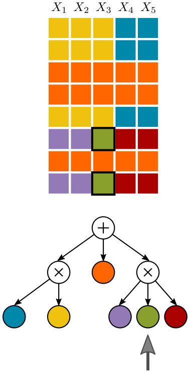


Gens and Domingos, "Learning the Structure of Sum-Product Networks", 2013

Learning the structure from data

"Recursive Data Slicing" — LearnSPN

Single variable

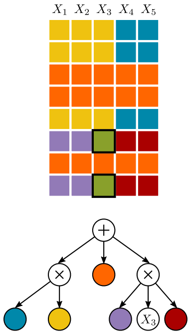


Gens and Domingos, "Learning the Structure of Sum-Product Networks", 2013

Learning the structure from data

"Recursive Data Slicing" — LearnSPN

Single variable $\longrightarrow$ ***leaf***

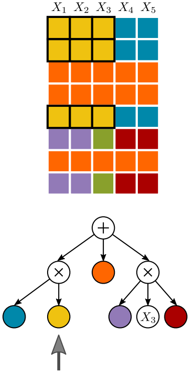


Gens and Domingos, "Learning the Structure of Sum-Product Networks", 2013

Learning the structure from data

"Recursive Data Slicing" — LearnSPN

Try to find independent groups of random variables



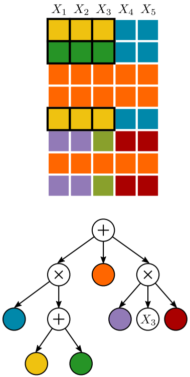
Gens and Domingos, "Learning the Structure of Sum-Product Networks", 2013

Learning the structure from data

“Recursive Data Slicing” — LearnSPN

Try to find independent groups of random variables

Fail $\longrightarrow$ cluster $\longrightarrow$ ***sum node***



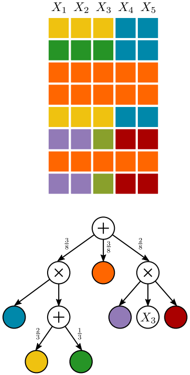
Gens and Domingos, “Learning the Structure of Sum-Product Networks”, 2013

Learning the structure from data

"Recursive Data Slicing" — LearnSPN

⇒ Continue until no further leaf can be expanded.

⇒ Clustering ratios also deliver (initial) parameters.

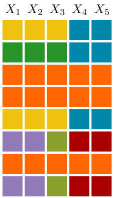


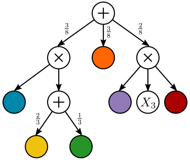
Gens and Domingos, "Learning the Structure of Sum-Product Networks", 2013

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





Gens and Domingos, "Learning the Structure of Sum-Product Networks", 2013

LearnSPN

Variants

- **ID-SPN** *[Rooshenas et al. 2014]*
- **LearnSPN-b/T/B** *[Vergari et al. 2015]*
- for **heterogeneous data** *[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]*

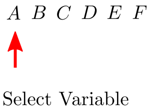
Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks *[Rahman et al. 2014]*

$A \quad B \quad C \quad D \quad E \quad F$

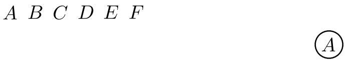
Structure Learning + MAP (determinism)

“Recursive conditioning” — Cutset Networks *[Rahman et al. 2014]*



Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks *[Rahman et al. 2014]*

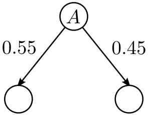


Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks *[Rahman et al. 2014]*

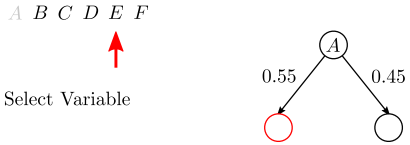
$A \quad B \quad C \quad D \quad E \quad F$

Split states



Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks *[Rahman et al. 2014]*

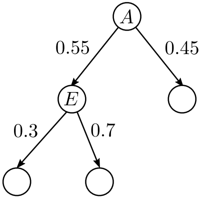


Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks *[Rahman et al. 2014]*

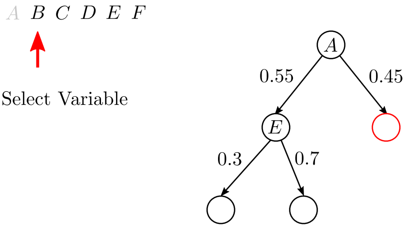
$A \;\; B \;\; C \;\; D \;\; E \;\; F$

Split states



Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks *[Rahman et al. 2014]*

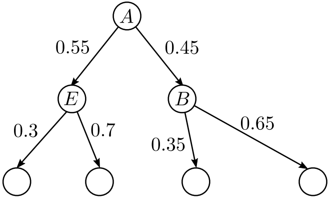


Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks *[Rahman et al. 2014]*

$A \;\; B \;\; C \;\; D \;\; E \;\; F$

Split states

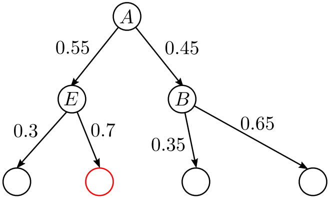


Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks *[Rahman et al. 2014]*

$A \;\; B \;\; C \;\; D \;\; E \;\; F$

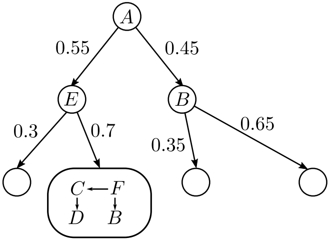
Stop → learn Chow-Liu



Structure Learning + MAP (determinism)

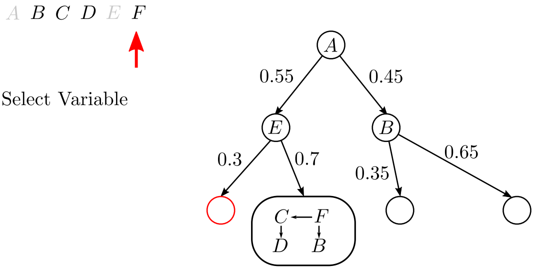
"Recursive conditioning" — Cutset Networks *[Rahman et al. 2014]*

$A \quad B \quad C \quad D \quad E \quad F$



Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks *[Rahman et al. 2014]*

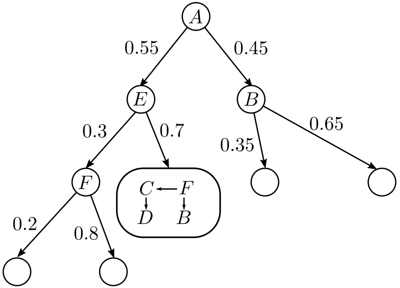


Structure Learning + MAP (determinism)

“Recursive conditioning” — Cutset Networks *[Rahman et al. 2014]*

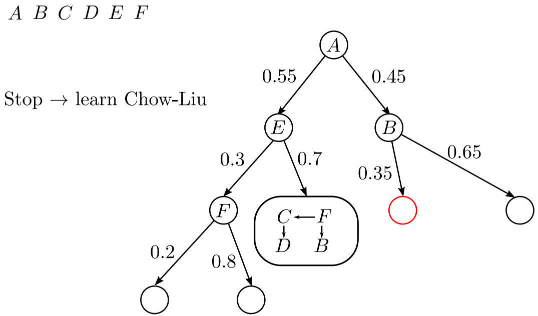
$A \;\; B \;\; C \;\; D \;\; E \;\; F$

Split states



Structure Learning + MAP (determinism)

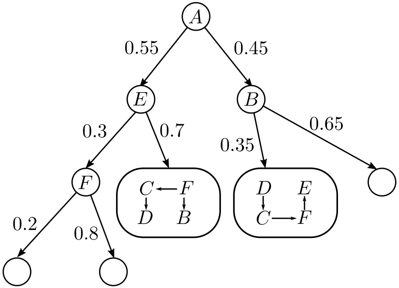
"Recursive conditioning" — Cutset Networks *[Rahman et al. 2014]*



Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks *[Rahman et al. 2014]*

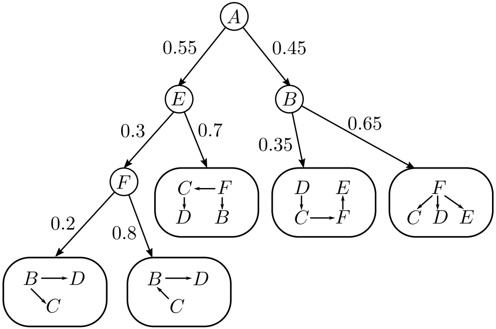
$A \quad B \quad C \quad D \quad E \quad F$



Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks *[Rahman et al. 2014]*

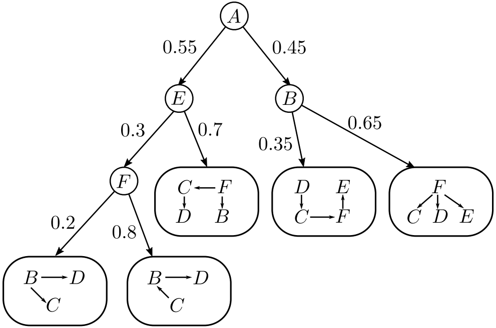
...and so on.



Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks *[Rahman et al. 2014]*

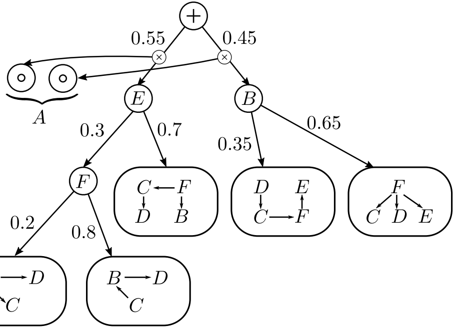
Convert into PC...



Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks *[Rahman et al. 2014]*

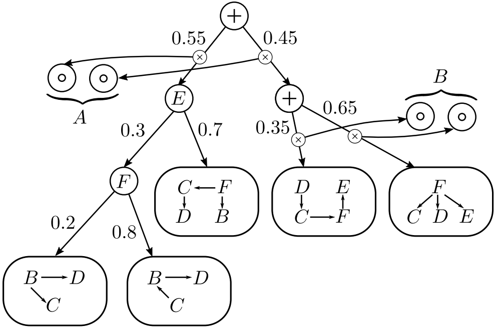
Convert into PC...



Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks *[Rahman et al. 2014]*

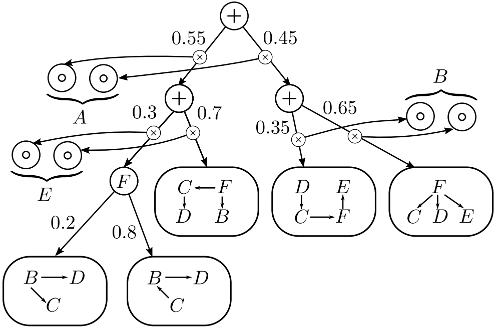
Convert into PC...



Structure Learning + MAP (determinism)

"Recursive conditioning" — Cutset Networks *[Rahman et al. 2014]*

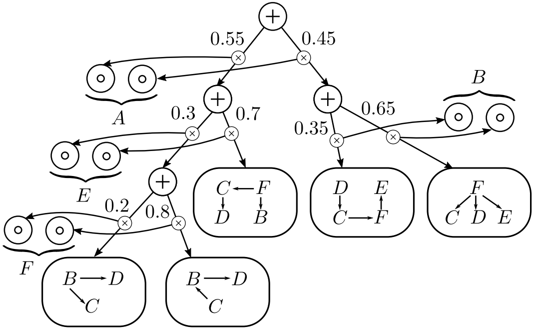
Convert into PC...



Structure Learning + MAP (determinism)

“Recursive conditioning” — Cutset Networks *[Rahman et al. 2014]*

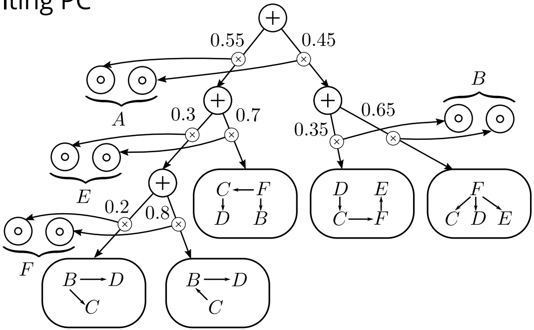
Convert into PC...



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. 2015b]*
- Boosted CNets *[Rahman et al. 2016]*
- Randomized CNets, Bagging *[Di Mauro et al. 2017]*

Further Algorithms for Structure Learning

Variants

- Greedy discrete optimization
  [Lowd et al. 2008; Peharz et al. 2014; Liang et al. 2017a; Dang et al. 2020]
- Randomized structures *[Di Mauro et al. 2017; Peharz et al. 2019b]*
- Ensembles, Bagging *[Di Mauro et al. 2015a,b],* Boosting *[Rahman et al. 2016]*

Learning probabilistic circuits

	Parameters	Structure
Generative	***deterministic*** closed-form MLE *[Kisa et al. 2014b; Peharz et al. 2014]* ***non-deterministic*** EM *[Poon et al. 2011; Peharz 2015; Zhao et al. 2016b]* SGD *[Sharir et al. 2016; Peharz et al. 2019b]* Bayesian *[Jaini et al. 2016; Rashwan et al. 2016]* *[Zhao et al. 2016a; 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; Dang et al. 2020]* ***random*** RAT-SPNs *[Peharz et al. 2019b]* XCNet *[Di Mauro et al. 2017]*
Discriminative	**?**	**?**

EVI inference : density estimation

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

Learning probabilistic circuits

	Parameters	Structure
Generative	***deterministic*** closed-form MLE *[Kisa et al. 2014b; Peharz et al. 2014]* ***non-deterministic*** EM *[Poon et al. 2011; Peharz 2015; Zhao et al. 2016b]* SGD *[Sharir et al. 2016; Peharz et al. 2019b]* Bayesian *[Jaini et al. 2016; Rashwan et al. 2016]* *[Zhao et al. 2016a; 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; Dang et al. 2020]* ***random*** RAT-SPNs *[Peharz et al. 2019b]* 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. 2019b]*	***greedy*** top-down *[Shao et al. 2019]* ***hill climbing*** *[Rooshenas et al. 2016; Liang et al. 2019]*

Advanced Representations

Tractability to other semi-rings

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

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

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

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

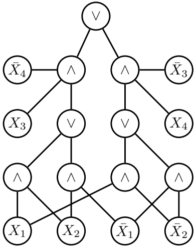
$\Longrightarrow$ *Algebraic model counting [Kimmig et al. 2017], Semi-ring programming [Belle et al. 2016]*

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

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

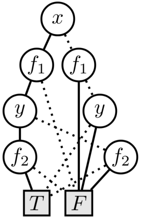
$\Longrightarrow$ *logical circuits!*

Logical circuits



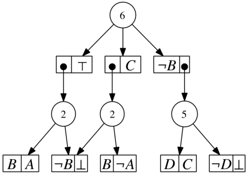
s/d-D/NNFs

[Darwiche et al. 2002a]



O/BDDs

[Bryant 1986]



SDDs

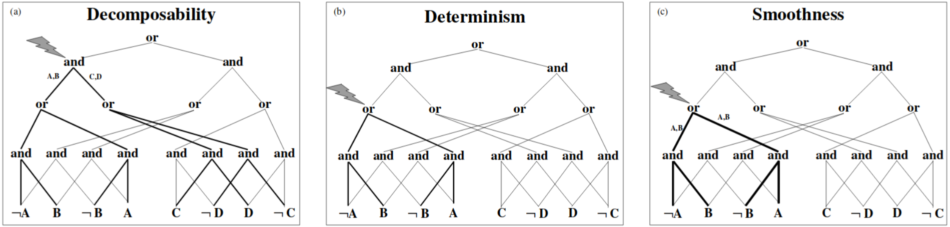
[Darwiche 2011]

Logical circuits are compact representations for boolean functions...

Logical circuits

structural properties

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

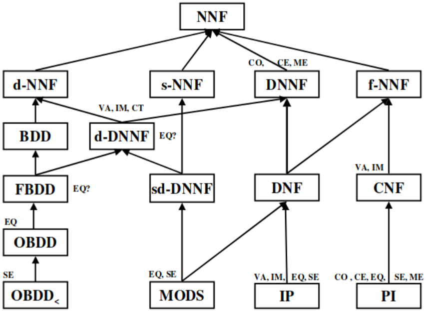


Darwiche and Marquis, “A knowledge compilation map”, 2002

Logical circuits

a knowledge compilation map

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



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

Logical circuits

connection to probabilistic circuits through WMC

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

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

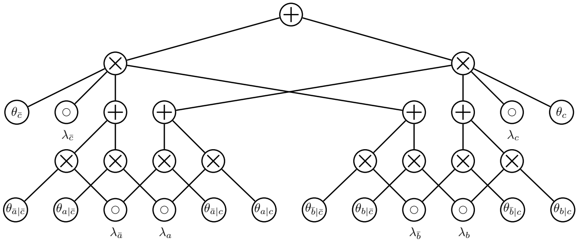
- Probabilistic inference by WMC:
  1. Encode probabilistic model as WMC formula Δ
  2. Compile Δ into a logical circuit (e.g. d-DNNF, OBDD, SDD, etc.)
  3. Tractable MAR/CON by tractable WMC on circuit
  4. Answer complex queries tractably by enforcing more structural properties

Logical circuits

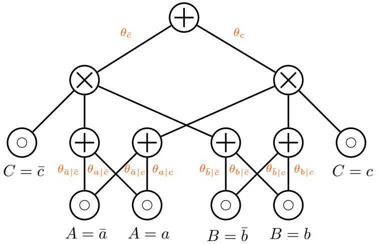
connection to probabilistic circuits through WMC

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

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



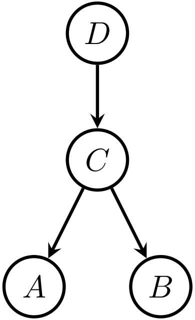
Compiled circuit of WMC encoding



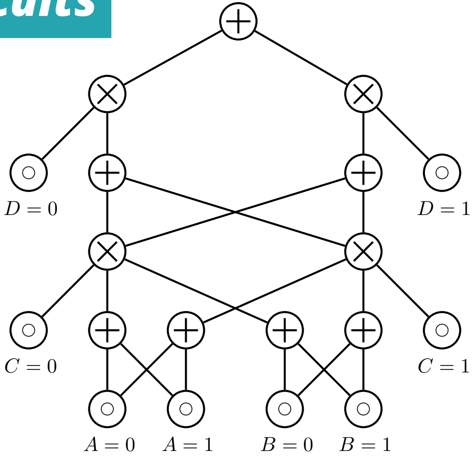
Equivalent probabilistic circuit

From BN trees to circuits

via compilation



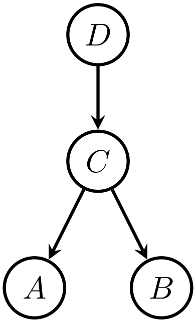
$\rightarrow$



From BN trees to circuits

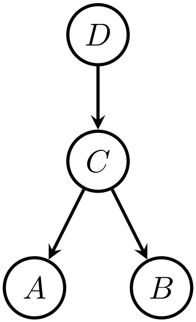
via compilation

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

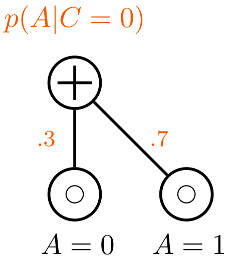


From BN trees to circuits

via compilation

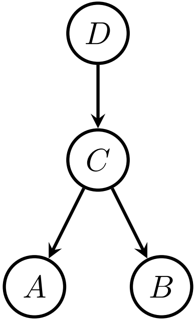


...compile a leaf CPT

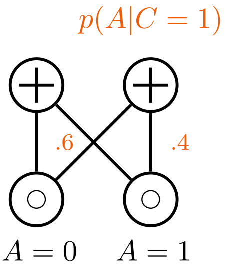


From BN trees to circuits

via compilation

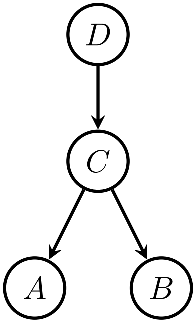


...compile a leaf CPT

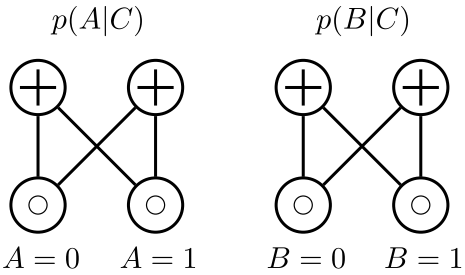


From BN trees to circuits

via compilation

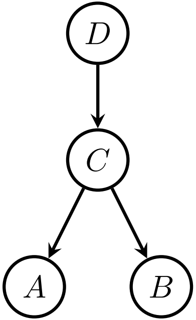


...compile a leaf CPT...for all leaves...

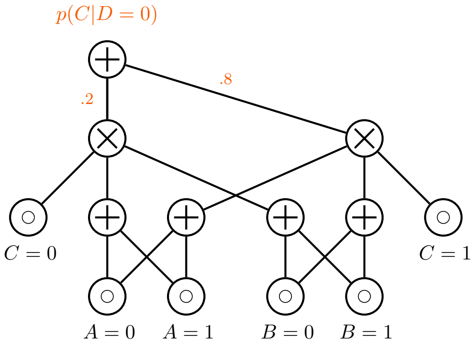


From BN trees to circuits

via compilation

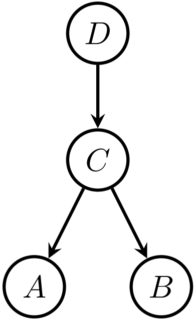


...and recurse over parents...

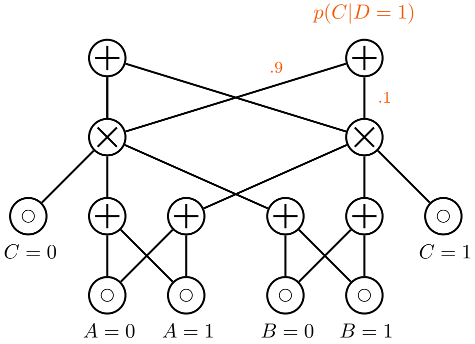


From BN trees to circuits

via compilation

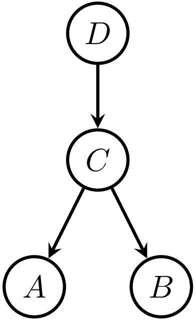


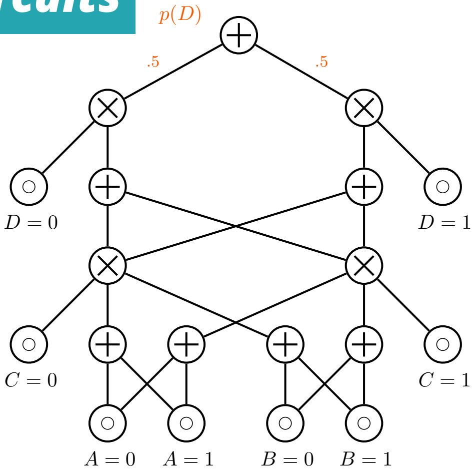
...while reusing previously compiled nodes!...



From BN trees to circuits

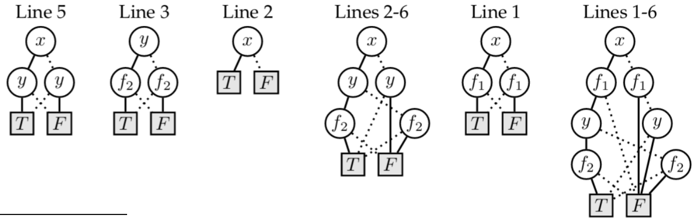
via compilation





Compilation: probabilistic programming

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



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

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

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

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

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

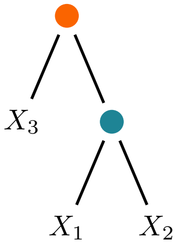
Smooth ∨ *decomposable* ∨ *deterministic* ∨ *structured decomposable* *PCs*?

	smooth	*dec.*	det.	str.dec.
Arithmetic Circuits (ACs) *[Darwiche 2003]*	✔	✔	✔	✘
Sum-Product Networks (SPNs) *[Poon et al. 2011]*	✔	✔	✘	✘
Cutset Networks (CNets) *[Rahman et al. 2014]*	✔	✔	✔	✘
Probabilistic Decision Graphs *[Jaeger 2004]*	✔	✔	✔	✔
PSDDs *[Kisa et al. 2014a]*	✔	✔	✔	✔
AndOrGraphs *[Dechter et al. 2007]*	✔	✔	✔	✔

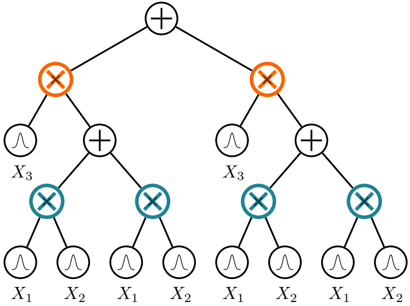
Structured decomposability

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

$\Rightarrow$ *stronger requirement than decomposability*



vtree

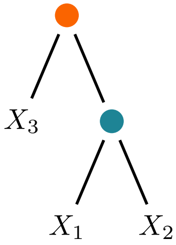


structured decomposable circuit

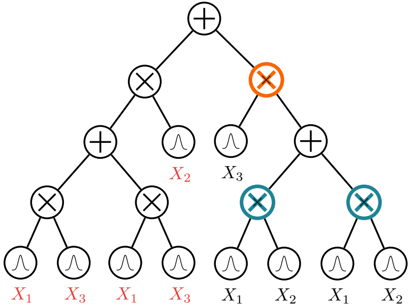
Structured decomposability

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

$\Longrightarrow$ *stronger requirement than decomposability*



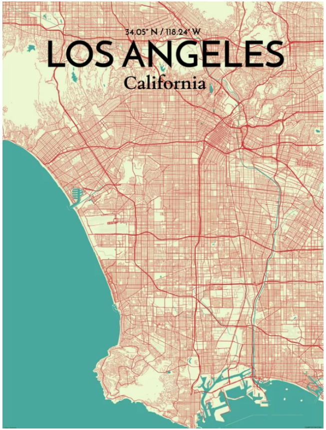
vtree



***non* structured decomposable circuit**

Probability of logical events

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



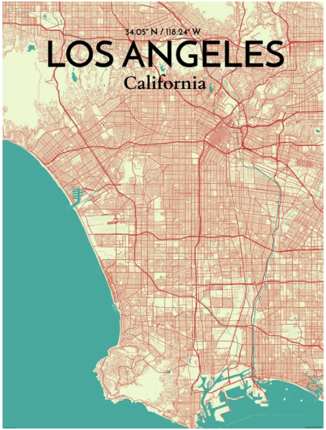
© fineartamerica.com

Probability of logical events

$\mathbf{q}_8$: *What is the probability of having a traffic jam on my route to campus?*

$$\mathbf{q}_8(\mathbf{m}) = p_{\mathbf{m}}(\bigvee_{i \in \text{route}} \text{Jam}_{\text{Str } i})$$

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



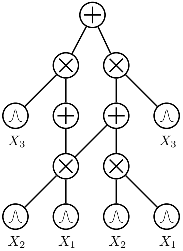
© fineartamerica.com

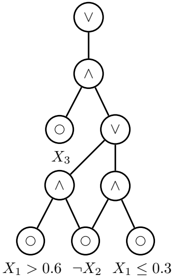
Smoothness + structured decomp. = tractable PR

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**



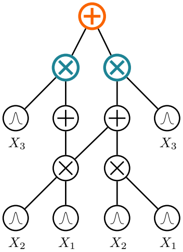


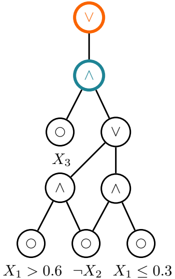
Smoothness + **structured decomp.** = **tractable PR**

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

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

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



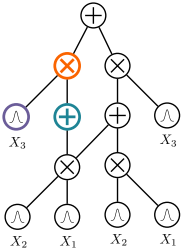


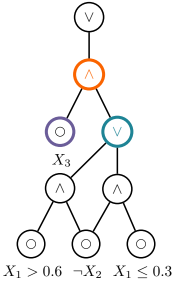
Smoothness + structured decomp. = tractable PR

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

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

$\Longrightarrow$ *probabilities decompose into simpler ones*





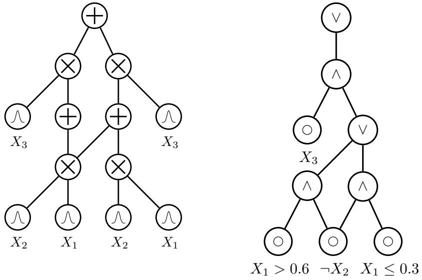
Smoothness + structured decomp. = tractable PR

To compute $p(\alpha)$:

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

  $\Longrightarrow$ *cache the values!*

- feedforward evaluation (bottom-up)



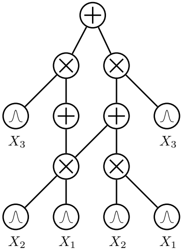
Smoothness + structured decomp. = tractable PR

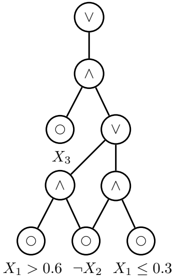
To compute $p(\alpha)$:

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

  $\Longrightarrow$ *cache the values!*

- feedforward evaluation (bottom-up)





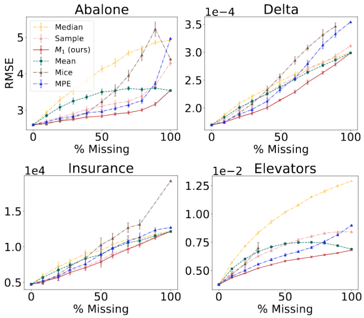
structured decomposability = tractable...

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

For the "right" vtree

- 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. 2019b]*

ADV inference : expected predictions



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

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

$\Longrightarrow$ **PR** *if* $\boldsymbol{f}$ *is a logical formula*

$\Longrightarrow$ *missing values at test time*

$\Longrightarrow$ *exploratory classifier analysis*

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

ADV inference in Julia with Juice.jl

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

q_8: *How different is the insurance costs between smokers and non smokers?*

```
groups = make_observations([["!smoker"], ["smoker"]])
exps, _ = Expectation(pc, rc, groups);
println("Smoker     : \$ $(exps[2])");
println("Non-Smoker: \$ $(exps[1])");
println("Difference: \$ $(exps[2] - exps[1])");
Smoker     : $ 31355.32630488978
Non-Smoker: $ 8741.747258310648
Difference: $ 22613.57904657913
```

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

ADV inference in Julia with Juice.jl

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

q_9: *Is the predictive model biased by gender?*

```julia
groups = make_observations([["male"], ["female"]])
exps, _ = Expectation(pc, rc, groups);
println("Female  : \$ $(exps[2])");
println("Male    : \$ $(exps[1])");
println("Diff    : \$ $(exps[2] - exps[1])");
Female  : $ 14170.125469335406
Male    : $ 13196.548926381849
Diff    : $ 973.5765429535568
```

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

Stay tuned for...

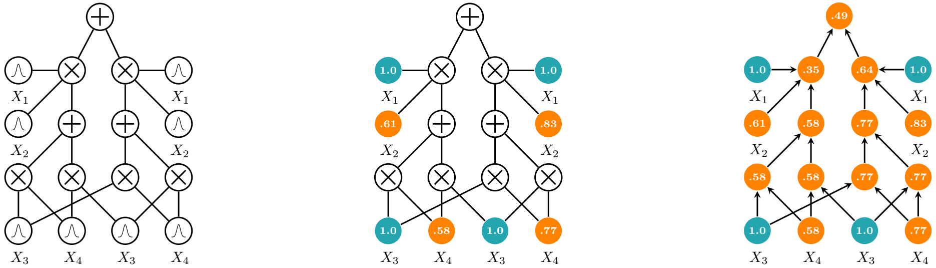
Next:

1. *How precise is the characterization of tractable circuits by structural properties?* $\Longrightarrow$ *necessary conditions*
2. *How do structural constraints affect the circuit sizes?* $\Longrightarrow$ *succinctness analysis*

After: *Conclusions!*

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

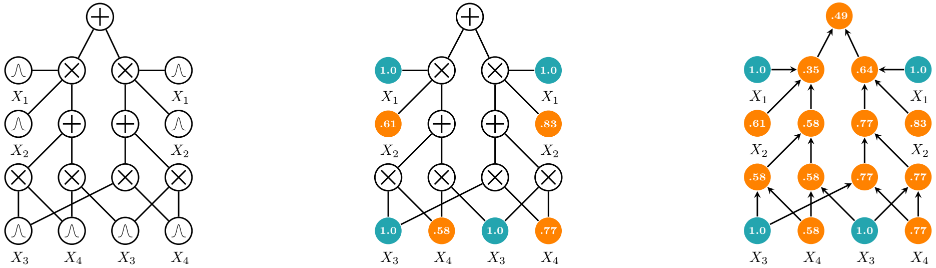
Recall: Smoothness and decomposability allow tractable computation of marginal queries.



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

Recall: Smoothness and decomposability allow tractable computation of marginal queries.

⇒ *Are these properties necessary?*

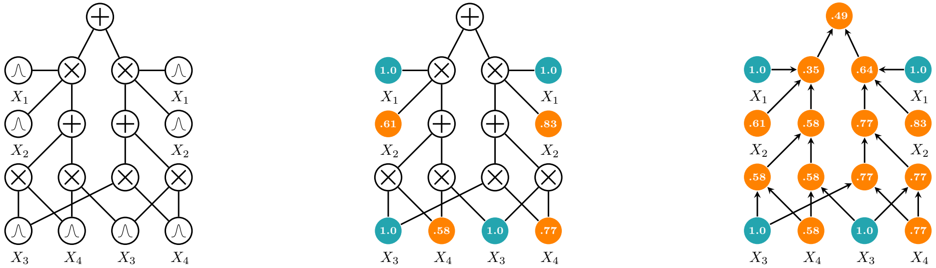


Smoothness + decomposability = tractable MAR

Recall: Smoothness and decomposability allow tractable computation of marginal queries.

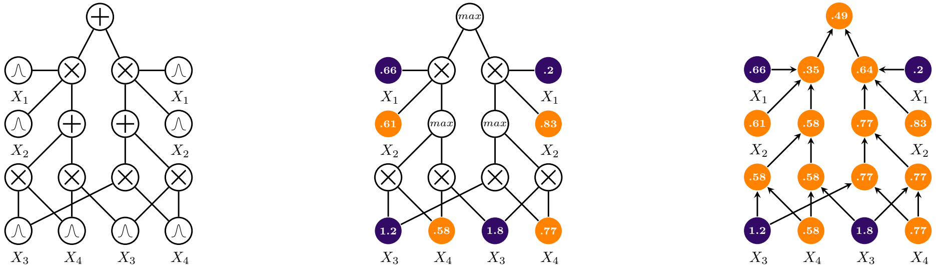
$\Rightarrow$ *Are these properties necessary?*

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



Determinism + decomposability = tractable MAP

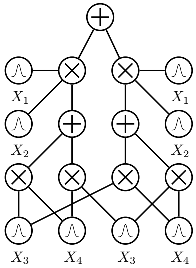
Recall: Determinism and decomposability allow tractable computation of MAP queries.

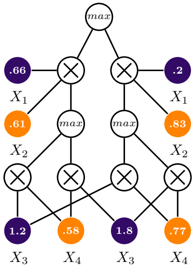


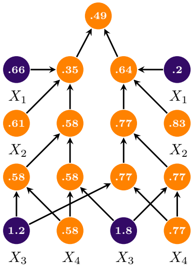
Determinism + **decomposability** = **tractable MAP**

Recall: Determinism and decomposability allow tractable computation of MAP queries.

⇒ *However, decomposability is not necessary!*





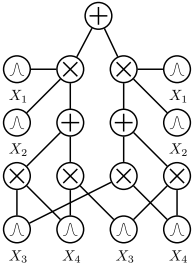


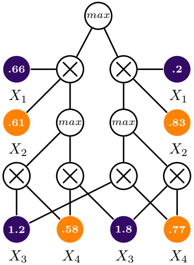
Determinism + **decomposability** = **tractable MAP**

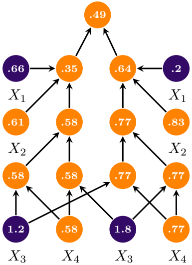
Recall: Determinism and decomposability allow tractable computation of MAP queries.

⇒ *However, decomposability is not necessary!*

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



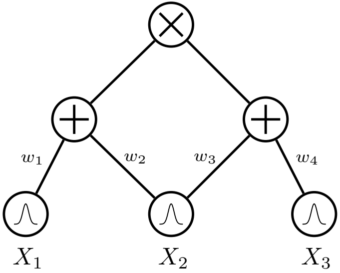




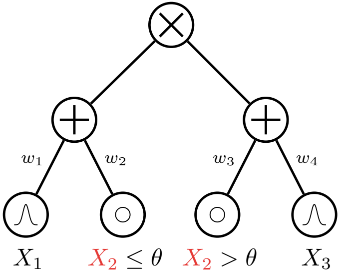
Consistency

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

$\Rightarrow$ *decomposability implies consistency*



consistent circuit



inconsistent circuit

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

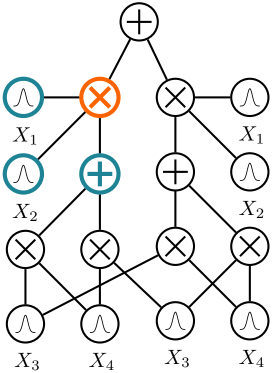
Determinism + consistency = tractable MAP

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

$$\max_{\mathbf{q}} p(\mathbf{q}, \mathbf{e}) = \max_{\mathbf{q}_{\mathbf{x}}, \mathbf{q}_{\mathbf{y}}} p(\mathbf{q}_{\mathbf{x}}, \mathbf{e}_{\mathbf{x}}, \mathbf{q}_{\mathbf{y}}, \mathbf{e}_{\mathbf{y}})$$

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

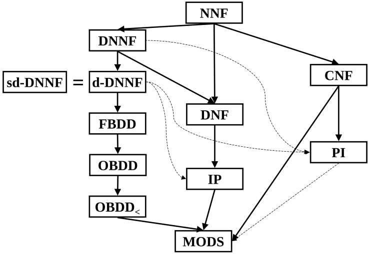
$\Rightarrow$ *solving optimization independently*



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*

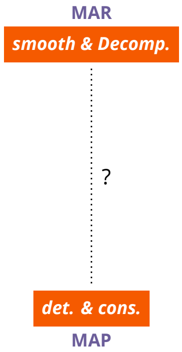
Expressive efficiency of circuits

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* $|m_1| \leq |\text{poly}(m_2)|$.

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

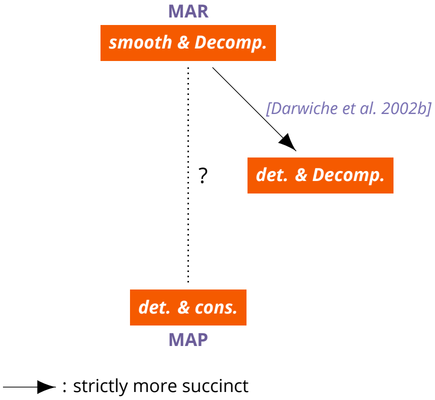
$\Rightarrow$ strictly more succinct iff $\mathcal{M}_1 \leq \mathcal{M}_2$ and $\mathcal{M}_1 \not\geq \mathcal{M}_2$

Expressive efficiency of circuits



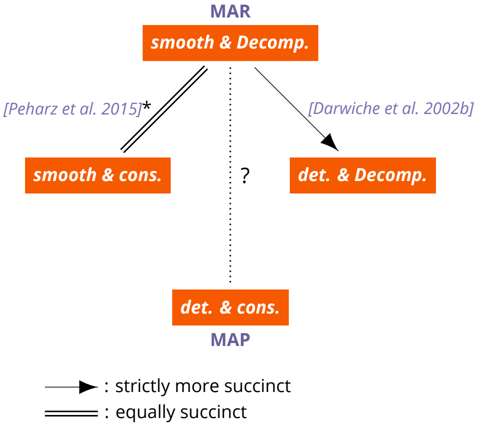
Are smooth & decomposable circuits as succinct as deterministic & consistent ones, or vice versa?

Expressive efficiency of circuits



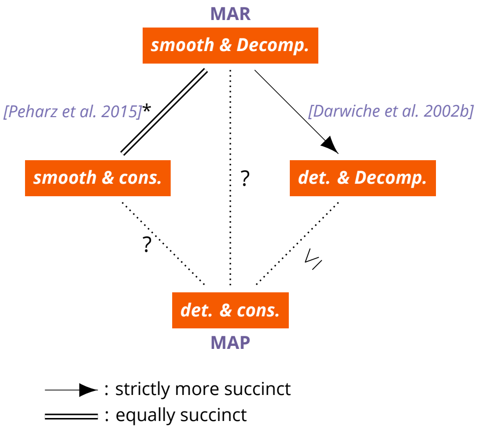
- Smooth & decomposable circuits strictly more succinct than deterministic & decomposable ones
- Smooth & consistent circuits are equally succinct as smooth & decomposable ones

Expressive efficiency of circuits



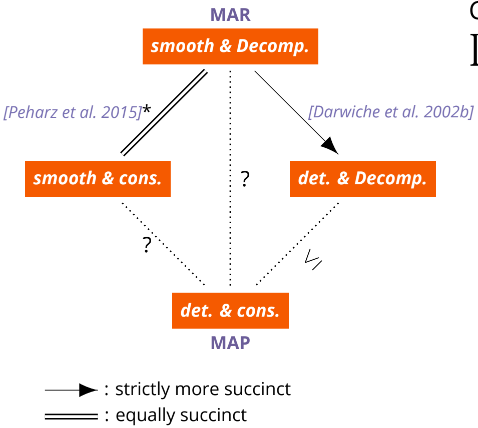
- Smooth & decomposable circuits strictly more succinct than deterministic & decomposable ones
- Smooth & consistent circuits are equally succinct as smooth & decomposable ones

Expressive efficiency of circuits



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

Expressive efficiency of circuits

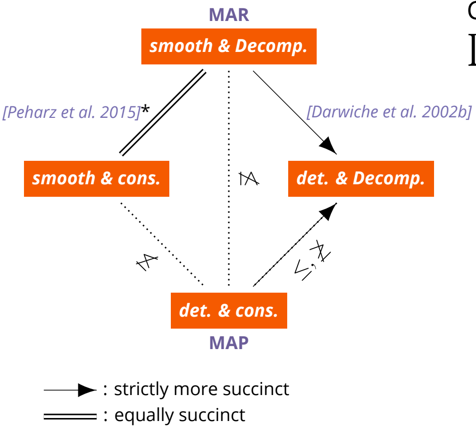


Consider following circuit over Boolean variables:

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

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

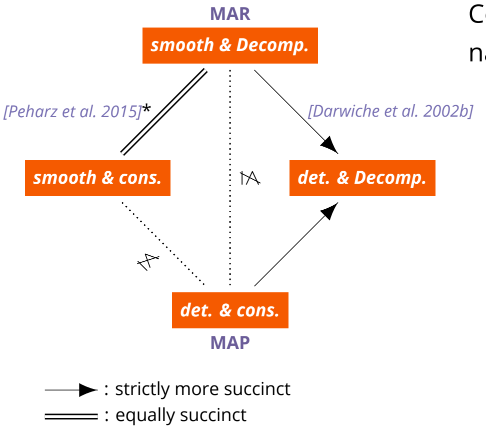
Expressive efficiency of circuits



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

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

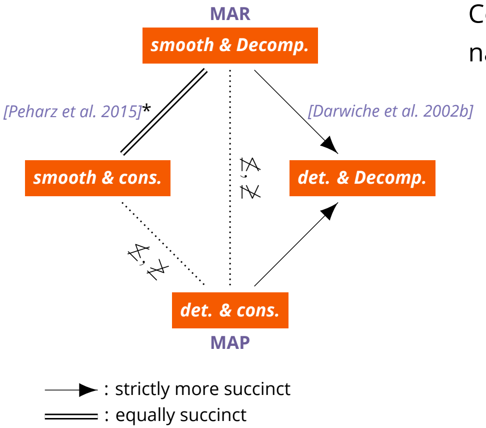
Expressive efficiency of circuits



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

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

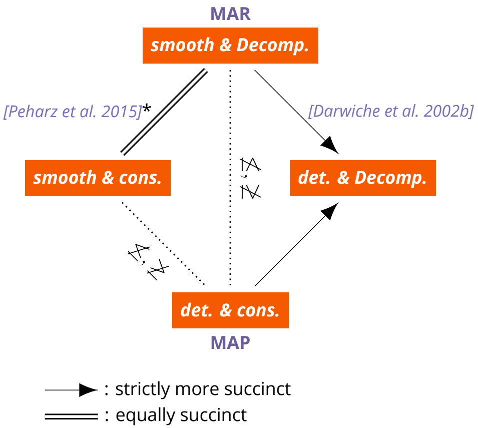
Expressive efficiency of circuits



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

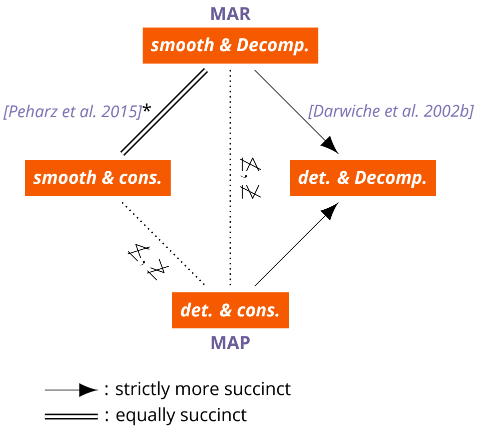
- Linear-size smooth and decomposable circuit
- MAP of $p(\mathbf{X})$ solves marginal MAP of $p(\mathbf{X}, C)$ which is NP-hard *[de Campos 2011]*
  $\Rightarrow$ **no tractable circuit for MAP!**

Expressive efficiency of circuits



- Neither smooth & decomposable nor deterministic & consistent circuits are more succinct than the other!
  ⇒ *Choose tractable circuit family based on your query*
- More theoretical questions remaining
  ⇒ *"Complete the map"*

Expressive efficiency of circuits



- 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

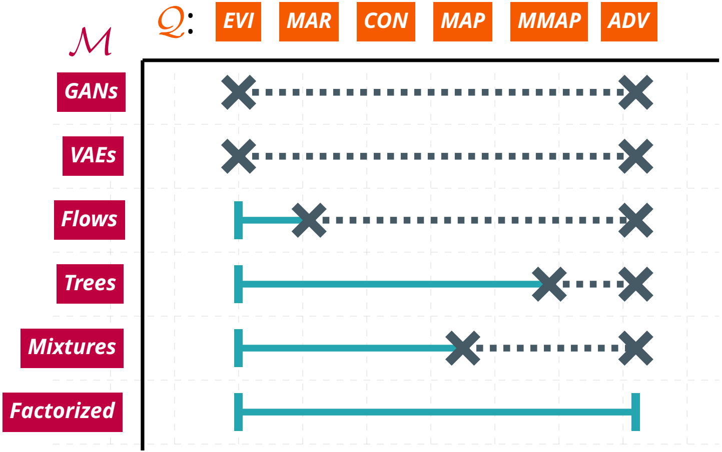
a unified framework for tractable probabilistic modeling

Learning circuits

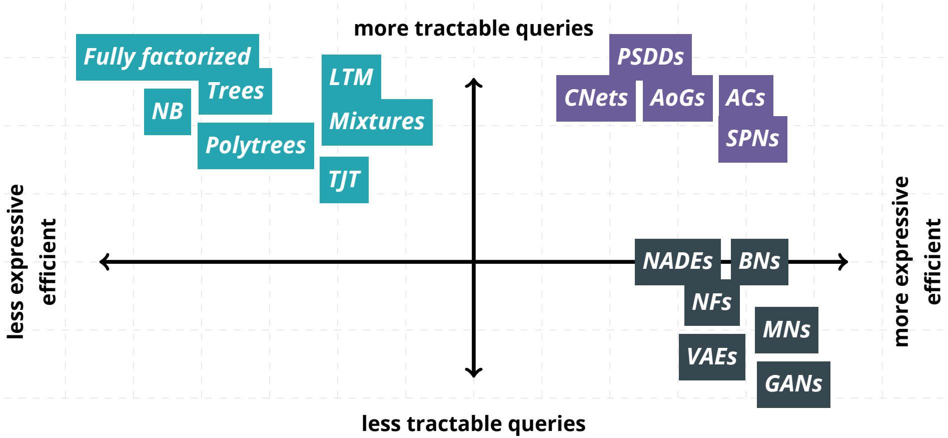
learning their structure and parameters from data

Advanced representations

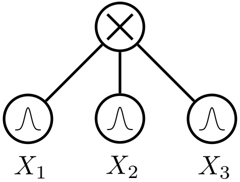
tracing the boundaries of tractability and connections to other formalisms

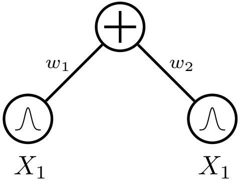


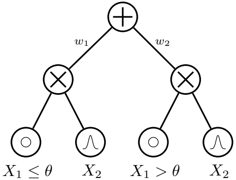
takeaway #1: tractability is a spectrum



takeaway #2: you can be both tractable and expressive







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

Challenge #1

scaling tractable learning

Learn tractable models

on **millions of datapoints**

and **thousands of features**

in tractable time!

Challenge #2

deep theoretical understanding

Trace a precise picture

of the ***whole tractabile spectrum***

and ***complete the map of succintness***!

Challenge #3

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/ECML20.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 $\Longrightarrow$ github.com/arranger1044/awesome-spn

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.
- Meilă, Marina and Michael I. Jordan (2000). “Learning with mixtures of trees”. In: *Journal of Machine Learning Research* 1, pp. 1–48.
- Bach, Francis R. and Michael I. Jordan (2001). “Thin Junction Trees”. In: *Advances in Neural Information Processing Systems 14*. MIT Press, pp. 569–576.
- Darwiche, Adnan (2001). “Recursive conditioning”. In: *Artificial Intelligence* 126.1-2, pp. 5–41.

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: *J.ACM*.
- Jaeger, Manfred (2004). “Probabilistic decision graphs—combining verification and AI techniques for probabilistic inference”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 12.supp01, pp. 19–42.
- Sang, Tian, Paul Beame, and Henry A Kautz (2005). “Performing Bayesian inference by weighted model counting”. In: *AAAI*. Vol. 5, pp. 475–481.
- Chavira, Mark, Adnan Darwiche, and Manfred Jaeger (2006). “Compiling relational Bayesian networks for exact inference”. In: *International Journal of Approximate Reasoning* 42.1-2, pp. 4–20.
- Jaeger, Manfred, Jens D Nielsen, and Tomi Silander (2006). “Learning probabilistic decision graphs”. In: *International Journal of Approximate Reasoning* 42.1-2, pp. 84–100.
- Park, James D and Adnan Darwiche (2006). “Complexity results and approximation strategies for MAP explanations”. In: *Journal of Artificial Intelligence Research* 21, pp. 101–133.

References III

- De Raedt, Luc, Angelika Kimmig, and Hannu Toivonen (2007). “ProbLog: A Probabilistic Prolog and Its Application in Link Discovery.”. In: *IJCAI*. Vol. 7. Hyderabad, pp. 2462–2467.
- Dechter, Rina and Robert Mateescu (2007). “AND/OR search spaces for graphical models”. In: *Artificial intelligence* 171.2-3, pp. 73–106.
- Marinescu, Radu and Rina Dechter (2007). “Best-first AND/OR search for 0/1 integer programming”. In: *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*. Springer, pp. 171–185.
- Riguzzi, Fabrizio (2007). “A top down interpreter for LPAD and CP-logic”. In: *Congress of the Italian Association for Artificial Intelligence*. Springer, pp. 109–120.
- 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.
- 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.

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, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). “Generative adversarial nets”. In: *Advances in neural information processing systems*, pp. 2672–2680.
- Kingma, Diederik P and Max Welling (2014). “Auto-Encoding Variational Bayes”. In: *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*. 2014.
- Kisa, Doga, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche (July 2014a). “Probabilistic sentential decision diagrams”. In: *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*. Vienna, Austria.
- — (July 2014b). “Probabilistic sentential decision diagrams”. In: *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*. Vienna, Austria. URL: `http://starai.cs.ucla.edu/papers/KisaKR14.pdf`.

References V

- Martens, James and Venkatesh Medabalimi (2014). “On the Expressive Efficiency of Sum Product Networks”. In: *CoRR* abs/1411.7717.
- Peharz, Robert, Robert Gens, and Pedro Domingos (2014). “Learning Selective Sum-Product Networks”. In: *Workshop on Learning Tractable Probabilistic Models*. LTPM.
- Rahman, Tahrima, Prasanna Kothalkar, and Vibhav Gogate (2014). “Cutset Networks: A Simple, Tractable, and Scalable Approach for Improving the Accuracy of Chow-Liu Trees”. In: *Machine Learning and Knowledge Discovery in Databases*. Vol. 8725. LNCS. Springer, pp. 630–645.
- Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). “Stochastic backprop. and approximate inference in deep generative models”. In: *arXiv preprint arXiv:1401.4082*.
- Rooshenas, Amirmohammad and Daniel Lowd (2014). “Learning Sum-Product Networks with Direct and Indirect Variable Interactions”. In: *Proceedings of ICML 2014*.
- Adel, Tameem, David Balduzzi, and Ali Ghodsi (2015). “Learning the Structure of Sum-Product Networks via an SVD-based Algorithm”. In: *Uncertainty in Artificial Intelligence*.
- Bekker, Jessa, Jesse Davis, Arthur Choi, Adnan Darwiche, and Guy Van den Broeck (2015). “Tractable Learning for Complex Probability Queries”. In: *Advances in Neural Information Processing Systems 28 (NIPS)*.
- 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)*.

References VI

- Choi, Arthur, Guy Van Den Broeck, and Adnan Darwiche (2015b). "Tractable Learning for Structured Probability Spaces: A Case Study in Learning Preference Distributions". In: *Proceedings of the 24th International Conference on Artificial Intelligence*. IJCAI'15. Buenos Aires, Argentina: AAAI Press, pp. 2861–2868. ISBN: 978-1-57735-738-4. URL: http://dl.acm.org/citation.cfm?id=2832581.2832649.
- Dennis, Aaron and Dan Ventura (2015). "Greedy Structure Search for Sum-product Networks". In: *IJCAI'15*. Buenos Aires, Argentina: AAAI Press, pp. 932–938. ISBN: 978-1-57735-738-4.
- Di Mauro, Nicola, Antonio Vergari, and Teresa M.A. Basile (2015a). "Learning Bayesian Random Cutset Forests". In: *Proceedings of ISMIS*. Springer, pp. 122–132.
- Di Mauro, Nicola, Antonio Vergari, and Floriana Esposito (2015b). "Learning Accurate Cutset Networks by Exploiting Decomposability". In: *Proceedings of AIXIA*. Springer, pp. 221–232.
- Fierens, Daan, Guy Van den Broeck, Joris Renkens, Dimitar Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt (May 2015). "Inference and Learning in Probabilistic Logic Programs using Weighted Boolean Formulas". In: *Theory and Practice of Logic Programming* 15 (03), pp. 358–401. ISSN: 1475-3081. DOI: 10.1017/S1471068414000076. URL: http://starai.cs.ucla.edu/papers/FierensTPLP15.pdf.
- Germain, Mathieu, Karol Gregor, Iain Murray, and Hugo Larochelle (2015). "MADE: Masked Autoencoder for Distribution Estimation". In: *CoRR* abs/1502.03509.
- Peharz, Robert (2015). "Foundations of Sum-Product Networks for Probabilistic Modeling". PhD thesis. Graz University of Technology, SPSC.
- Peharz, Robert, Sebastian Tschiatschek, Franz Pernkopf, and Pedro Domingos (2015). "On Theoretical Properties of Sum-Product Networks". In: *The Journal of Machine Learning Research*.
- Vergari, Antonio, Nicola Di Mauro, and Floriana Esposito (2015). "Simplifying, Regularizing and Strengthening Sum-Product Network Structure Learning". In: *ECML-PKDD 2015*.

References VII

- Vlasselaer, Jonas, Guy Van den Broeck, Angelika Kimmig, Wannes Meert, and Luc De Raedt (2015). "Anytime Inference in Probabilistic Logic Programs with Tp-compilation". In: *Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI)*. URL: `http://starai.cs.ucla.edu/papers/VlasselaerIJCAI15.pdf`.
- Belle, Vaishak and Luc De Raedt (2016). "Semiring Programming: A Framework for Search, Inference and Learning". In: *arXiv preprint arXiv:1609.06954*.
- 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.
- Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2016). "Density estimation using real nvp". In: *arXiv preprint arXiv:1605.08803*.
- Friesen, Abram L and Pedro Domingos (2016). "Submodular Sum-product Networks for Scene Understanding". In:
- Jaini, Priyank, Abdullah Rashwan, Han Zhao, Yue Liu, Ershad Banijamali, Zhitang Chen, and Pascal Poupart (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, Robert Gens, Franz Pernkopf, and Pedro M. Domingos (2016). "On the Latent Variable Interpretation in Sum-Product Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP, Issue 99. URL: `http://arxiv.org/abs/1601.06180`.
- Pronobis, A. and R. P. N. Rao (2016). "Learning Deep Generative Spatial Models for Mobile Robots". In: *ArXiv e-prints*. arXiv: `1610.02627 [cs.RO]`.

References VIII

- Rahman, Tahrima and Vibhav Gogate (2016). “Learning Ensembles of Cutset Networks”. In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI’16. Phoenix, Arizona: AAAI Press, pp. 3301–3307. URL: `http://dl.acm.org/citation.cfm?id=3016100.3016365`.
- Rashwan, Abdullah, Han Zhao, and Pascal Poupart (2016). “Online and Distributed Bayesian Moment Matching for Parameter Learning in Sum-Product Networks”. In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pp. 1469–1477.
- Rooshenas, Amirmohammad and Daniel Lowd (2016). “Discriminative Structure Learning of Arithmetic Circuits”. In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pp. 1506–1514.
- Sguerra, Bruno Massoni and Fabio G Cozman (2016). “Image classification using sum-product networks for autonomous flight of micro aerial vehicles”. In: *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*. IEEE, pp. 139–144.
- Sharir, Or, Ronen Tamari, Nadav Cohen, and Amnon Shashua (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, Wannes Meert, Guy Van den Broeck, and Luc De Raedt (Mar. 2016). “Exploiting Local and Repeated Structure in Dynamic Bayesian Networks”. In: *Artificial Intelligence* 232, pp. 43 –53. ISSN: 0004-3702. DOI: `10.1016/j.artint.2015.12.001`.
- Yuan, Zehuan, Hao Wang, Limin Wang, Tong Lu, Shivakumara Palaiahnakote, and Chew Lim Tan (2016). “Modeling spatial layout for scene image understanding via a novel multiscale sum-product network”. In: *Expert Systems with Applications* 63, pp. 231–240.

References IX

- Zhao, Han, Tameem Adel, Geoff Gordon, and Brandon Amos (2016a). “Collapsed Variational Inference for Sum-Product Networks”. In: *In Proceedings of the 33rd International Conference on Machine Learning*. Vol. 48.
- Zhao, Han, Pascal Poupart, and Geoffrey J Gordon (2016b). “A Unified Approach for Learning the Parameters of Sum-Product Networks”. In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., pp. 433–441.
- Alemi, Alexander A, Ben Poole, Ian Fischer, Joshua V Dillon, Rif A Saurous, and Kevin Murphy (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, Antonio Vergari, Teresa M. A. Basile, and Floriana Esposito (2017). “Fast and Accurate Density Estimation with Extremely Randomized Cutset Networks”. In: *ECML-PKDD 2017*.
- Kimmig, Angelika, Guy Van den Broeck, and Luc De Raedt (2017). “Algebraic model counting”. In: *Journal of Applied Logic* 22, pp. 46–62.
- Liang, Yitao, Jessa Bekker, and Guy Van den Broeck (2017a). “Learning the structure of probabilistic sentential decision diagrams”. In: *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*.
- Liang, Yitao and Guy Van den Broeck (Aug. 2017b). “Towards Compact Interpretable Models: Shrinking of Learned Probabilistic Sentential Decision Diagrams”. In: *IJCAI 2017 Workshop on Explainable Artificial Intelligence (XAI)*. URL: `http://starai.cs.ucla.edu/papers/LiangXAI17.pdf`.

References X

- Papamakarios, George, Theo Pavlakou, and Iain Murray (2017). "Masked autoregressive flow for density estimation". In: *Advances in Neural Information Processing Systems*, pp. 2338–2347.
- 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.
- Van den Broeck, Guy and Dan Suciu (Aug. 2017). *Query Processing on Probabilistic Data: A Survey*. Foundations and Trends in Databases. Now Publishers. DOI: `10.1561/1900000052`. URL: `http://starai.cs.ucla.edu/papers/VdBFTDB17.pdf`.
- Butz, Cory J, Jhonatan S Oliveira, André E Santos, André L Teixeira, Pascal Poupart, and Agastya Kalra (2018). "An Empirical Study of Methods for SPN Learning and Inference". In: *International Conference on Probabilistic Graphical Models*, pp. 49–60.
- Choi, YooJung and Guy Van den Broeck (2018). "On robust trimming of Bayesian network classifiers". In: *arXiv preprint arXiv:1805.11243*.
- Di Mauro, Nicola, Floriana Esposito, Fabrizio Giuseppe Ventola, and Antonio Vergari (2018). "Sum-Product Network structure learning by efficient product nodes discovery". In: *Intelligenza Artificiale* 12.2, pp. 143–159.
- Jaini, Priyank, Amur Ghose, and Pascal Poupart (2018). "Prometheus: Directly Learning Acyclic Directed Graph Structures for Sum-Product Networks". In: *International Conference on Probabilistic Graphical Models*, pp. 181–192.

References XI

- Molina, Alejandro, Antonio Vergari, Nicola Di Mauro, Sriraam Natarajan, Floriana Esposito, and Kristian Kersting (2018). “Mixed Sum-Product Networks: A Deep Architecture for Hybrid Domains”. In: *AAAI*.
- 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, Mehdi SM Sajjadi, Antonio Vergari, Michael Black, and Bernhard Schölkopf (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, YooJung Choi, Yitao Liang, Antonio Vergari, and Guy Van den Broeck (2019a). “On Tractable Computation of Expected Predictions”. In: *Advances in Neural Information Processing Systems*, pp. 11167–11178.
- Khosravi, Pasha, Yitao Liang, YooJung Choi, and Guy Van den Broeck (2019b). “What to Expect of Classifiers? Reasoning about Logistic Regression with Missing Features”. In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*.

References XII

- Khosravi, Pasha, Yitao Liang, YooJung Choi, and Guy Van den Broeck (2019c). “What to Expect of Classifiers? Reasoning about Logistic Regression with Missing Features”. In: *arXiv preprint arXiv:1903.01620*.
- Kossen, Jannik, Karl Stelzner, Marcel Hussing, Claas Voelcker, and Kristian Kersting (2019). “Structured Object-Aware Physics Prediction for Video Modeling and Planning”. In: *arXiv preprint arXiv:1910.02425*.
- Liang, Yitao and Guy Van den Broeck (2019). “Learning Logistic Circuits”. In: *Proceedings of the 33rd Conference on Artificial Intelligence (AAAI)*.
- Molina, Alejandro, Antonio Vergari, Karl Stelzner, Robert Peharz, Pranav Subramani, Nicola Di Mauro, Pascal Poupart, and Kristian Kersting (2019). “SPFlow: An easy and extensible library for deep probabilistic learning using sum-product networks”. In: *arXiv preprint arXiv:1901.03704*.
- Peharz, Robert, Antonio Vergari, Karl Stelzner, Alejandro Molina, Xiaoting Shao, Martin Trapp, Kristian Kersting, and Zoubin Ghahramani (2019a). “Random sum-product networks: A simple but effective approach to probabilistic deep learning”. In: *Proceedings of UAI*.
- Peharz, Robert, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Xiaoting Shao, Kristian Kersting, and Zoubin Ghahramani (2019b). “Random Sum-Product Networks: A Simple and Effective Approach to Probabilistic Deep Learning”. In: *Uncertainty in Artificial Intelligence*.
- Shao, Xiaoting, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, and Kristian Kersting (2019). “Conditional Sum-Product Networks: Imposing Structure on Deep Probabilistic Architectures”. In: *arXiv preprint arXiv:1905.08550*.
- Shih, Andy, Guy Van den Broeck, Paul Beame, and Antoine Amarilli (2019). “Smoothing Structured Decomposable Circuits”. In: *arXiv preprint arXiv:1906.00311*.

References XIII

- Stelzner, Karl, Robert Peharz, and Kristian Kersting (2019). “Faster Attend-Infer-Repeat with Tractable Probabilistic Models”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, pp. 5966–5975. URL: `http://proceedings.mlr.press/v97/stelzner19a.html`.
- Tan, Ping Liang and Robert Peharz (2019). “Hierarchical Decompositional Mixtures of Variational Autoencoders”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, pp. 6115–6124. URL: `http://proceedings.mlr.press/v97/tan19b.html`.
- Trapp, Martin, Robert Peharz, Hong Ge, Franz Pernkopf, and Zoubin Ghahramani (2019). “Bayesian Learning of Sum-Product Networks”. In: *Advances in neural information processing systems (NeurIPS)*.
- Vergari, Antonio, Alejandro Molina, Robert Peharz, Zoubin Ghahramani, Kristian Kersting, and Isabel Valera (2019). “Automatic Bayesian density analysis”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33, pp. 5207–5215.
- Dang, Meihua, Antonio Vergari, and Guy Van den Broeck (2020). “Strudel: Learning Structured-Decomposable Probabilistic Circuits”. In: *The 10th International Conference on Probabilistic Graphical Models (PGM)*.
- Peharz, Robert, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van den Broeck, Kristian Kersting, and Zoubin Ghahramani (2020). “Einsum Networks: Fast and Scalable Learning of Tractable Probabilistic Circuits”. In: *ICML*.
- Zhang, Honghua, Steven Holtzen, and Guy Van den Broeck (2020). “On the Relationship Between Probabilistic Circuits and Determinantal Point Processes”. In: *Proceedings of the 36th Conference on Uncertainty in Aritifical Intelligence (UAI)*.