Probabilistic Circuits

Antonio Vergari
University of California, Los Angeles

Robert Peharz
TU Eindhoven

YooJung Choi
University of California, Los Angeles

Guy Van den Broeck
University of California, Los Angeles

September 14th, 2020 - Ghent, Belgium - ECML-PKDD 2020
The Alphabet Soup of probabilistic models
Intractable and tractable models
tractability is a spectrum
Expressive models without compromises
a **unifying framework** for tractable models
Why tractable inference?
or expressiveness vs tractability
Why tractable inference?

or expressiveness vs tractability

Probabilistic circuits

a unified framework for tractable probabilistic modeling
Why tractable inference?
or expressiveness vs tractability

Probabilistic circuits
a unified framework for tractable probabilistic modeling

Learning circuits
learning their structure and parameters from data
Why tractable inference?

or expressiveness vs tractability

Probabilistic circuits

a unified framework for tractable probabilistic modeling

Learning circuits

learning their structure and parameters from data

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.?
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?
**Why probabilistic inference?**

**q₁**: What is the probability that today is a Monday and there is a traffic jam on Westwood Blvd.?

**q₂**: 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?”

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?”

\[ q_1(m_1)? \Rightarrow p_{m_1}(Y \mid X) \approx \]

...by fitting predictive models!
“What is the most likely time to see a traffic jam at Sunset Blvd.?"
“What is the probability of a traffic jam on Westwood Blvd. on Monday?”

\[ q_1(m_1)? \rightarrow q_3(m_?)? \rightarrow p_{m_?}(Y) \]

…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.?
Why probabilistic inference?

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

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

$q_1(m) = p_m(\text{Day} = \text{Mon}, \text{Jam}_{\text{Wwood}} = 1)$
Why probabilistic inference?

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

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

$q_1(m) = p_m(\text{Day} = \text{Mon}, \text{Jam}_{\text{Westwood}} = 1)$

$\Rightarrow$ marginals
Why probabilistic inference?

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

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

$q_2(m) = \arg\max_d p_m(\text{Day} = d \land \bigvee_{i \in \text{route}} \text{Jam}_{\text{Stri}})$
Why probabilistic inference?

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

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

$$q_2(m) = \arg\max_d p_m(\text{Day} = d \land \bigvee_{i \in \text{route}} \text{Jam}_{\text{Str}i})$$

⇒ marginals + MAP + logical events
Tractable Probabilistic Inference

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

A class of queries $Q$ is tractable on a family of probabilistic models $M$ iff for any query $q \in Q$ and model $m \in M$

*exactly* computing $q(m)$ runs in time $O(poly(|m|))$.

$\Rightarrow$ often poly will in fact be linear!
Tractable Probabilistic Inference

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

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

$\Rightarrow$ Note: if $M$ is compact in the number of random variables $X$, that is, $|m| \in O(\text{poly}(|X|))$, then query time is $O(\text{poly}(|X|))$. 
Tractable Probabilistic Inference

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

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

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

$\Rightarrow$ Why exactness? Highest guarantee possible!
1. What are classes of queries?
2. Are my favorite models tractable?
3. Are tractable models expressive?

We introduce probabilistic circuits as a unified framework for tractable probabilistic modeling.
tractable bands
q₃: What is the probability that today is a Monday at 12.00 and there is a traffic jam only on Westwood Blvd.?
**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.?*

\[ X = \{\text{Day}, \text{Time}, \text{Jam}_{\text{Westwood}}, \text{Jam}_{\text{Str1}}, \ldots, \text{Jam}_{\text{StrN}}\} \]

\[ q_3(m) = p_m(X = \{\text{Mon, 12.00, 1, 0, \ldots, 0}\}) \]
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.?

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

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

...fundamental in maximum likelihood learning

$\theta_{m}^{\text{MLE}} = \arg\max_\theta \prod_{x \in D} p_m(x; \theta)$
Generative Adversarial Networks

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

Goodfellow et al., “Generative adversarial nets”, 2014
Generative Adversarial Networks

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

- no explicit likelihood!

  \( \Rightarrow \) adversarial training instead of MLE

  \( \Rightarrow \) no tractable EVI

- good sample quality

  \( \Rightarrow \) but lots of samples needed for MC

- unstable training

  \( \Rightarrow \) mode collapse

Goodfellow et al., “Generative adversarial nets”, 2014
tractable bands
Variational Autoencoders

\[ p_\theta(x) = \int p_\theta(x \mid z)p(z)dz \]

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

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

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

\[ p_X(x) = p_Z(f^{-1}(x)) \left| \det \left( \frac{\delta f^{-1}}{\delta x} \right) \right| \]

- 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

\[ p_X(x) = p_Z(f^{-1}(x)) \left| \det \left( \frac{\delta f^{-1}}{\delta x} \right) \right| \]

- 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

\[ p_X(x) = p_Z(f^{-1}(x)) \left| \det \left( \frac{\delta f^{-1}}{\delta x} \right) \right| \]

- 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.?
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.?

$q_1(m) = p_m(\text{Day} = \text{Mon}, \text{Jam}_{\text{Westwood}} = 1)$
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.?

$q_1(m) = p_m(\text{Day} = \text{Mon}, \text{Jam}_{\text{Westwood}} = 1)$

General: $p_m(e) = \int p_m(e, H) dH$

where $E \subset X$, $H = X \setminus E$

© fineartamerica.com
Marginal queries (MAR)

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

$q_1(m) = p_m(\text{Day} = \text{Mon}, \text{Jam}_{\text{Westwood}} = 1)$

tractable MAR \implies \text{tractable conditional queries (CON)}:

$$p_m(q \mid e) = \frac{p_m(q, e)}{p_m(e)}$$
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
\[ p_X(x) = p_Z(f^{-1}(x)) \left| \det \left( \frac{\delta f^{-1}}{\delta x} \right) \right| \]

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

\[ \Rightarrow \quad \text{tractable EVI queries!} \]
Normalizing flows

\[ p_X(x) = p_Z(f^{-1}(x)) \left| \text{det} \left( \frac{\delta f^{-1}}{\delta x} \right) \right| \]

- an explicit likelihood!
- ...plus structured Jacobians \( \Rightarrow \) tractable EVI queries!
- **MAR is generally intractable**: we cannot easily integrate over \( f \)
  \( \Rightarrow \) 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

\[
\Rightarrow \quad [\text{Cooper 1990; Roth 1996}]
\]

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

\[
\Rightarrow \quad [\text{Dagum et al. 1993; Roth 1996}]
\]
**Why? Treewidth!**

**Treewidth:**
Informally, how tree-like is the graphical model $m$? Formally, the minimum width of any tree-decomposition of $m$.

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

[Dechter 1998; Koller et al. 2009].

$\Rightarrow$ 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. $\approx 20$), 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 $\text{Pa}_{X_i}$.

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

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

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

---

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 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 \textit{categorical latent variable} \( Z \) with \( k \) values

\( \Rightarrow \) increased expressiveness
Expressiveness and efficiency

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

⇒ mixture of Gaussians can approximate any distribution!

---

Expressiveness and efficiency

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

⇒ mixture of Gaussians can approximate any distribution!

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

⇒ but how many components does a Gaussian mixture need?

---

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?

⇒  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)

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

$q_5(m) = \arg\max_j p_m(j_1, j_2, \ldots | \text{Day} = M, \text{Time} = 9)$
Maximum A Posteriori (MAP)

aka Most Probable Explanation (MPE)

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

q_5(m) = \arg\max_j p_m(j_1, j_2, \ldots | \text{Day} = M, \text{Time} = 9)

General: \arg\max_q p_m(q | e)

where \( Q \cup E = X \)
**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!

\[
\max_q p_m(q \mid e) = \max_q \sum_z p_m(q, z \mid e)
\]
\[
\neq \sum_z \max_q p_m(q, z \mid e)
\]
MAP inference: image inpainting

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

---

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?
Marginal MAP (MMAP)

aka Bayesian Network MAP

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

$q_6(m) = \arg\max_j p_m(j_1, j_2, \ldots | \text{Time} = 9)$
Marginal MAP (MMAP) aka Bayesian Network MAP

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

$q_6(m) = \arg\max_j p_m(j_1, j_2, \ldots | \text{Time}=9)$

General: $\arg\max_q p_m(q | e) = \arg\max_q \sum_h p_m(q, h | e)$

where $Q \cup H \cup E = X$
Marginal MAP (MMAP)

aka Bayesian Network MAP

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

$q_6(m) = \text{argmax}_j p_m(j_1, j_2, \ldots | \text{Time} = 9)$

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

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

$\Rightarrow$ 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?
Advanced queries

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

$q_2(m) = \arg\max_d p_m(Day = d \land \bigvee_{i \in \text{route}} \text{Jam}_{str_i})$

⇒ marginals + MAP + logical events

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?

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?

⇒ counts + group comparison

Advanced queries

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

q7: What is the probability of seeing more traffic jams in Westwood than Hollywood?

and more:

- expected classification agreement

- expected predictions [Khosravi et al. 2019c]

tractable bands
<table>
<thead>
<tr>
<th>Model Class</th>
<th>EVI</th>
<th>MAR</th>
<th>CON</th>
<th>MAP</th>
<th>MMAP</th>
<th>ADV</th>
</tr>
</thead>
<tbody>
<tr>
<td>GANs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VAEs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flows</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trees</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mixtures</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**tractable bands**
Fully factorized models

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

\[
p(x) = \prod_{i=1}^{n} p(x_i)
\]

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

⇒ but definitely not expressive...
tractable bands
Expressive models are not very tractable...
and tractable ones are not very expressive...
probabilistic circuits are at the "sweet spot"
Probabilistic Circuits
A probabilistic circuit $\mathcal{C}$ over variables $\mathbf{X}$ is a computational graph encoding a (possibly unnormalized) probability distribution $p(\mathbf{X})$. 

Probabilistic circuits
Probabilistic circuits

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

How can probabilistic circuits be learned?
**Distributions as computational graphs**

**Base case:** a single node encoding a distribution

⇒ e.g., Gaussian PDF continuous random variable
Base case: a single node encoding a distribution

\[ \neg X \]

\[ \Rightarrow \text{ e.g., indicators for } X \text{ or } \neg X \text{ for Boolean random variable} \]
Distributions as computational graphs

\[ x \xrightarrow{\wedge} p_X(x) \]

Simple distributions are tractable “black boxes” for:

- **EVI**: output \( p(x) \) (density or mass)
- **MAR**: output 1 (normalized) or \( Z \) (unnormalized)
- **MAP**: output the mode
Simple distributions are tractable “black boxes” for:

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

Divide and conquer complexity

\[ p(X_1, X_2, X_3) = p(X_1) \cdot p(X_2) \cdot p(X_3) \]

⇒ e.g. modeling a multivariate Gaussian with diagonal covariance matrix...
Factorizations are product nodes

Divide and conquer complexity

\[ p(X_1, X_2, X_3) = p(X_1) \cdot p(X_2) \cdot p(X_3) \]

\[ \Rightarrow \text{...with a product node over some univariate Gaussian distribution} \]
Factorizations are product nodes

Divide and conquer complexity

\[ p(x_1, x_2, x_3) = p(x_1) \cdot p(x_2) \cdot p(x_3) \]

⇒ feedforward evaluation
Factorizations are product nodes

Divide and conquer complexity

\[ p(x_1, x_2, x_3) = p(x_1) \cdot p(x_2) \cdot p(x_3) \]

\[ \Rightarrow \text{feedforward evaluation} \]
Mixtures
Enhance expressiveness

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

⇒ e.g. modeling a mixture of Gaussians...
Mixtures are sum nodes

Enhance expressiveness

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

⇒ ...as a weighted sum node over Gaussian input distributions
Mixtures are sum nodes

Enhance expressiveness

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

⇒ 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
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))

Probabilistic circuits are not PGMs!

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

<table>
<thead>
<tr>
<th>PGMs</th>
<th>Circuits</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nodes:</strong></td>
<td>random variables</td>
</tr>
<tr>
<td><strong>Edges:</strong></td>
<td>dependencies</td>
</tr>
<tr>
<td><strong>Inference:</strong></td>
<td>conditioning</td>
</tr>
<tr>
<td></td>
<td>elimination</td>
</tr>
<tr>
<td></td>
<td>message passing</td>
</tr>
</tbody>
</table>

⇒ 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!

\[ \text{structural constraints needed for tractability} \]
Which structural constraints ensure tractability?
**Decomposability**

A product node is decomposable if its children depend on disjoint sets of variables just like in factorization!

\[ X_1 \times X_2 \times X_3 \]

decomposable circuit

\[ X_1 \times X_1 \times X_3 \]

non-decomposable circuit

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

*aka completeness*

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

\[ \Rightarrow \] otherwise not accounting for some variables

\[ \begin{align*}
&w_1 \quad w_2 \\
\wedge &X_1 \\
\wedge &X_1
\end{align*} \]

**smooth circuit**

\[ \begin{align*}
&w_1 \quad w_2 \\
\wedge &X_1 \\
\wedge &X_1
\end{align*} \]

**non-smooth circuit**

\[ \begin{align*}
&w_1 \quad w_2 \\
\wedge &X_1 \\
\wedge &X_2
\end{align*} \]

\[ \Rightarrow \] smoothness can be easily enforced \[ \text{[Shih et al. 2019]} \]

Darwiche and Marquis, “A knowledge compilation map”, 2002
\textbf{Smoothness} + \textbf{decomposability} = \textbf{tractable MAR}

Computing arbitrary integrations (or summations)
\[ \Rightarrow \text{linear in circuit size!} \]

E.g., suppose we want to compute \( Z \):
\[
\int p(x) \, dx
\]
**Smoothness** + **decomposability** = **tractable MAR**

If \( p(x) = \sum_i w_i p_i(x) \), (smoothness):

\[
\int p(x) \, dx = \int \sum_i w_i p_i(x) \, dx = \sum_i w_i \int p_i(x) \, dx
\]

\( \Rightarrow \) integrals are “pushed down” to children
**Smoothness** + **decomposability** = **tractable MAR**

If \( p(x, y, z) = p(x)p(y)p(z) \), (decomposability):

\[
\int \int \int p(x, y, z) \, dx \, dy \, dz = \\
= \int \int \int p(x)p(y)p(z) \, dx \, dy \, dz = \\
= \int p(x) \, dx \int p(y) \, dy \int p(z) \, dz
\]

⇒ **integrals decompose into easier ones**
Smoothness + decomposability = tractable MAR

Forward pass evaluation for MAR

⇒ linear in circuit size!

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

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

⇒ for normalized leaf distributions: 1.0

- leafs over \( X_2 \) and \( X_4 \) output EVI

- feedforward evaluation (bottom-up)
**Smoothness** + **decomposability** = **tractable MAR**

Forward pass evaluation for MAR

⇒ *linear in circuit size!*

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

- Leafs over $X_1$ and $X_3$ output $Z_i = \int p(x_i) dx_i$
  ⇒ for normalized leaf distributions: 1.0

- Leafs over $X_2$ and $X_4$ output **EVI**

- Feedforward evaluation (bottom-up)
**Smoothness** + **decomposability** = **tractable MAR**

Forward pass evaluation for MAR

⇒ *linear in circuit size!*

E.g. to compute $p(x_2, x_4)$:
- leafs over $X_1$ and $X_3$ output $Z_i = \int p(x_i) dx_i$
  ⇒ *for normalized leaf distributions:* 1.0
- leafs over $X_2$ and $X_4$ output **EVI**
- feedforward evaluation (bottom-up)
**Smoothness** + **decomposability** = **tractable CON**

Analogously, for arbitrary conditional queries:

\[
p(q | e) = \frac{p(q, e)}{p(e)}
\]

1. evaluate \( p(q, e) \)  \(\Rightarrow\)  **one feedforward pass**
2. evaluate \( p(e) \)  \(\Rightarrow\)  **another feedforward pass**

\(\Rightarrow\)  **...still linear in circuit size!**
Tractable MAR on PCs (Einsum Networks)

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

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

---

Pronobis et al., “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:

\[
\max_q p(q \mid e)
\]
Smoothness + decomposability = tractable MAP

We cannot decompose bottom-up a MAP query:

$$\max_q p(q \mid e)$$

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

$$\max_q \sum_i w_i p_i(q, e) = \max_q \sum_z p(q, z, e) \neq \sum_z \max_q p(q, z, e)$$

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

aka selectivity

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

$$\implies \text{e.g. if their distributions have disjoint support}$$

---

**deterministic circuit**

**non-deterministic circuit**

Darwiche and Marquis, “A knowledge compilation map”, 2002
Determinism + decomposability = tractable MAP

Computing maximization with arbitrary evidence $e$ \[\Rightarrow\] linear in circuit size!

E.g., suppose we want to compute:

$$\max_{q} p(q \mid e)$$
Determinism + decomposability = tractable MAP

If \( p(q, e) = \sum_i w_i p_i(q, e) = \max_i w_i p_i(q, e) \),
(deterministic sum node):

\[
\max_q p(q, e) = \max_q \sum_i w_i p_i(q, e) \\
= \max_i \max_q w_i p_i(q, e) \\
= \max_i \max_q w_i p_i(q, e)
\]

\( \Rightarrow \) one non-zero child term, thus sum is max
Determinism + decomposability = tractable MAP

If \( p(q, e) = p(q_x, e_x, q_y, e_y) = p(q_x, e_x)p(q_y, e_y) \)
(decomposable product node):

\[
\max_q p(q \mid e) = \max_q p(q, e) = \max_{q_x, q_y} p(q_x, e_x, q_y, e_y) = \max_{q_x} p(q_x, e_x) \cdot \max_{q_y} p(q_y, e_y)
\]
⇒ solving optimization independently
**Determinism** + **decomposability** = **tractable MAP**

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

![Circuit Diagram]
Determinism + decomposability = tractable MAP

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

E.g., for $\arg\max_{x_1, x_3} p(x_1, x_3 \mid x_2, x_4)$:

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 ⇒ still linear in circuit size!

E.g., for $\arg\max_{x_1, x_3} p(x_1, x_3 \mid x_2, x_4)$:
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 ⇒ still linear in circuit size!

E.g., for 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* ⇒ *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
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
Determinism + decomposability = tractable MMAP

Analogously, we could also do a MMAP query:

$$\max_{q} \sum_{z} p(q, z \mid e)$$
**Determinism** + **decomposability** = **tractable MMAP**

We *cannot* decompose a MMAP query!

\[
\max_q \sum_z p(q, z | e)
\]

we still have latent variables to marginalize...

We need more structural properties!

⇒ *more advanced queries in Part 4 later...*
where are probabilistic circuits?
more tractable queries

less tractable queries

tractability vs expressive efficiency
Low-treewidth PGMs

Tree, polytrees and Thin Junction trees can be turned into
- decomposable
- smooth
- deterministic circuits

Therefore they support tractable
- EVI
- MAR/CON
- MAP
Arithmetic Circuits (ACs)

ACs [Darwiche 2003] are
- decomposable
- smooth
- deterministic

They support tractable
- EVI
- MAR/CON
- MAP

⇒ parameters are attached to the leaves
⇒ ...but can be moved to the sum node edges [Rooshenas et al. 2014]

Lowd and Rooshenas, “Learning Markov Networks With Arithmetic Circuits”, 2013
SPNs [Poon et al. 2011] are decomposable, smooth, and deterministic. They support tractable EVI, MAR/CON, and MAP.

⇒ deterministic SPNs are also called selective [Peharz et al. 2014]
Cutset Networks (CNets)

CNets [Rahman et al. 2014] are decomposable, smooth, and deterministic. They support tractable EVI, MAR/CON, and MAP.

---


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, and deterministic. They support tractable EVI, MAR/CON, MAP, and Complex queries!

Kisa et al., “Probabilistic sentential decision diagrams”, 2014
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
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” leaves?

Concretely: Can all binary distributions that are tractable for MAR be written as smooth decomposable PCs with univariate leaves?
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” leaves?

Concretely: Can all binary distributions that are tractable for MAR be written as smooth decomposable PCs with univariate leaves?
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” leaves?

- Concretely: Can all binary distributions that are tractable for MAR be written as smooth decomposable PCs with univariate leaves?
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” leaves?

Concretely: Can all binary distributions that are tractable for MAR be written as smooth decomposable PCs with univariate leaves?

Almost! One possible exception are Determinantal Point Processes (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
# How expressive are probabilistic circuits?

**Density estimation benchmarks**

<table>
<thead>
<tr>
<th>dataset</th>
<th>best circuit</th>
<th>BN</th>
<th>MADE</th>
<th>VAE</th>
<th>dataset</th>
<th>best circuit</th>
<th>BN</th>
<th>MADE</th>
<th>VAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>nltcs</td>
<td><strong>-5.99</strong></td>
<td>-6.02</td>
<td>-6.04</td>
<td><strong>-5.99</strong></td>
<td>dna</td>
<td><strong>-79.88</strong></td>
<td>-80.65</td>
<td>-82.77</td>
<td>-94.56</td>
</tr>
<tr>
<td>msnbc</td>
<td><strong>-6.04</strong></td>
<td>-6.04</td>
<td>-6.06</td>
<td>-6.09</td>
<td>kosarek</td>
<td><strong>-10.52</strong></td>
<td>-10.83</td>
<td></td>
<td>-10.64</td>
</tr>
<tr>
<td>plants</td>
<td><strong>-11.84</strong></td>
<td>-12.65</td>
<td>-12.32</td>
<td>-12.34</td>
<td>book</td>
<td>-33.82</td>
<td>-36.41</td>
<td>-33.95</td>
<td><strong>-33.19</strong></td>
</tr>
<tr>
<td>audio</td>
<td>-39.39</td>
<td>-40.50</td>
<td>-38.95</td>
<td><strong>-38.67</strong></td>
<td>movie</td>
<td>-50.34</td>
<td>-54.37</td>
<td>-48.7</td>
<td><strong>-47.43</strong></td>
</tr>
<tr>
<td>jester</td>
<td>-51.29</td>
<td><strong>-51.07</strong></td>
<td>-52.23</td>
<td>-51.54</td>
<td>webkb</td>
<td>-149.20</td>
<td>-157.43</td>
<td>-149.59</td>
<td><strong>-146.9</strong></td>
</tr>
<tr>
<td>netflix</td>
<td>-55.71</td>
<td>-57.02</td>
<td>-55.16</td>
<td><strong>-54.73</strong></td>
<td>cr52</td>
<td>-81.87</td>
<td>-87.56</td>
<td>-82.80</td>
<td><strong>-81.33</strong></td>
</tr>
<tr>
<td>accidents</td>
<td>-26.89</td>
<td><strong>-26.32</strong></td>
<td>-26.42</td>
<td>-29.11</td>
<td>c20ng</td>
<td>-151.02</td>
<td>-158.95</td>
<td>-153.18</td>
<td><strong>-146.9</strong></td>
</tr>
<tr>
<td>retail</td>
<td><strong>-10.72</strong></td>
<td>-10.87</td>
<td>-10.81</td>
<td>-10.83</td>
<td>bbc</td>
<td><strong>-229.21</strong></td>
<td>-257.86</td>
<td>-242.40</td>
<td>-240.94</td>
</tr>
<tr>
<td>pumbs*</td>
<td>-22.15</td>
<td><strong>-21.72</strong></td>
<td>-22.3</td>
<td>-25.16</td>
<td>ad</td>
<td>-14.00</td>
<td>-18.35</td>
<td><strong>-13.65</strong></td>
<td>-18.81</td>
</tr>
</tbody>
</table>
Hybrid intractable + tractable EVI

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

\[ \rightarrow \text{decomposing a joint ELBO: better lower-bounds than a single VAE} \]
\[ \rightarrow \text{more expressive efficient and less data hungry} \]

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

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

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generative</td>
<td>?</td>
</tr>
<tr>
<td>Discriminative</td>
<td>?</td>
</tr>
</tbody>
</table>

**Table:**

- **Parameters**
  - Generative: ?
  - Discriminative: ?

- **Structure**
  - Generative: ?
  - Discriminative: ?
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, ...

How circuits are related to other tractable models?
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

\begin{align*}
X_1
\end{align*}

The simplest PC: a single input distribution $p_L$ with parameters $\theta$

$\Rightarrow$ maximum likelihood (ML) estimation over data $D$
Learning input distributions

As simple as tossing a coin

The simplest PC: a single input distribution $p_L$ with parameters $\theta$  

$\Rightarrow$ maximum likelihood (ML) estimation over data $D$

E.g. Bernoulli with parameter $\theta$

$$\hat{\theta}_{ML} = \frac{\sum_{x \in D} 1[x = 1] + \alpha}{|D| + 2\alpha}$$

$\Rightarrow$ Laplace smoothing
Learning input distributions

General case: still simple

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

\[ p_L(x) = h(x) \exp(T(x)^T \theta - A(\theta)) \]
Learning input distributions

General case: still simple

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

\[ p_L(x) = h(x) \exp(T(x)^T \theta - A(\theta)) \]

Where:
- \( A(\theta) \): log-normalizer
- \( h(x) \): base-measure
- \( T(x) \): sufficient statistics
- \( \theta \): natural parameters
Learning input distributions

General case: still simple

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

\[ p_L(x) = h(x) \exp(T(x)^T \theta - A(\theta)) \]

Where:
- \( A(\theta) \): log-normalizer
- \( h(x) \) base-measure
- \( T(x) \) sufficient statistics
- \( \theta \) natural parameters
- or \( \phi \) expectation parameters — 1:1 mapping with \( \theta \Rightarrow \theta = \theta(\phi) \)
Learning input distributions

General case: still simple

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

\[ p_L(x) = h(x) \exp(T(x)^T \theta - A(\theta)) \]

Maximum likelihood estimation is still "counting":

\[ \hat{\phi}_{ML} = \mathbb{E}_D[T(x)] = \frac{1}{|D|} \sum_{x \in D} T(x) \]

\[ \hat{\theta}_{ML} = \theta(\hat{\phi}_{ML}) \]
Recall that sum nodes represent *mixture models*:

\[
p_S(x) = \sum_{k=1}^{K} w_k p_{L_k}(x)
\]
Recall that sum nodes represent **latent variable models**:

\[
p_S(x) = \sum_{k=1}^{K} p(Z = k)p(x | Z = k)
\]
**Expectation-Maximization (EM)**

*Learning latent variable models: the EM recipe*

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

Given: \( p(X, Z) \) where \( X \) observed, \( Z \) missing at random.

\[
\theta^{new} \leftarrow \arg \max_\theta \mathbb{E}_{p(z | x; \theta^{old})} \left[ \log p(X, Z; \theta) \right]
\]
Expectation-Maximization for mixtures

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

ML if \( Z \) was observed:

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

\( Z \) is unobserved—but we have \( p(Z = k \mid x) \propto w_k L_k(x) \).

\[ w^{\text{new}}_k = \frac{\sum_{x \in \mathcal{D}} p(Z = k \mid x)}{|\mathcal{D}|} \quad \phi^{\text{new}}_k = \frac{\sum_{x, z \in \mathcal{D}} p(Z = k \mid x) T(x)}{\sum_{z \in \mathcal{D}} p(Z = k \mid x)} \]
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

\[ P(Z = 1) \quad \text{and} \quad P(Z = 2) \]

\[ w_1 \quad \text{and} \quad w_2 \]

[PEHARZ ET AL. 2016]
**Expectation-Maximization for PCs**

\[ P(Z = 1) \]

\[ P(Z = 2) \]

\[ P(X \mid Z = 1) \]

\[ P(X \mid Z = 2) \]

[Peharz et al. 2016]
Expectation-Maximization for PCs

\[ P(Z = 1) \]
\[ P(Z = 2) \]

\[ P(X \mid Z = 1) \]
\[ P(X \mid Z = 2) \]
Expectation-Maximization for PCs

\[ P(Z = 1) \quad P(Z = 2) \]

\[ P(X \mid Z = 1) \quad P(X \mid Z = 2) \]

\[ ctx = 1 \]

[Peharz et al. 2016]
Expectation-Maximization for PCs

\[ P(Z = 1 \mid ctx = 1) \]

\[ P(Z = 2 \mid ctx = 1) \]

\[ P(X \mid Z = 1, ctx = 1) \]

\[ P(X \mid Z = 2, ctx = 1) \]

[Pecharz et al. 2016]
For learning, we need to know for each sum $S$:

1. Is $S$ reached ($ctx = ?$)
2. Which child does it select ($Z_S = ?$)
For learning, we need to know for each sum $S$:

1. Is $S$ reached ($ctx = ?$)
2. Which child does it select ($Z_S = ?$)
For learning, we need to know for each sum $S$:

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

We can infer it: $p(context, Z_S | x)$
Expectation-Maximization

Tractable MAR (smooth, decomposable)

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

Darwiche, “A Differential Approach to Inference in Bayesian Networks”, 2003
**Expectation-Maximization**

Tractable MAR (smooth, decomposable)

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

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

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

---

Darwiche, “A Differential Approach to Inference in Bayesian Networks”, 2003
Expectation-Maximization

Tractable MAR (smooth, decomposable)

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

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

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

⇒ This also works with missing values in \( x \! \! \)!

Darwiche, “A Differential Approach to Inference in Bayesian Networks”, 2003
Expectation-Maximization

Tractable MAR (smooth, decomposable)

\[
\begin{align*}
\w_{i,j}^{new} & \leftarrow \frac{\sum_{x \in D} p[ctx_i = 1, Z_i = j \mid x; w^{old}]}{\sum_{x \in D} p[ctx_i = 1 \mid x; w^{old}]} \\
\end{align*}
\]

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

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

\[\Rightarrow \text{ Similar updates for leaves, when in exponential family.}\]

Darwiche, “A Differential Approach to Inference in Bayesian Networks”, 2003
**Expectation-Maximization**

*Tractable MAR (smooth, decomposable)*

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

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

\[ p[ctx_i = 1, Z_i = j | x; w^{\text{old}}] = \frac{1}{p(x)} \frac{\partial p(x)}{\partial S_i(x)} N_j(x) w_{i,j}^{\text{old}} \]

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

Darwiche, “A Differential Approach to Inference in Bayesian Networks”, 2003

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

```python
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

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

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...
For example, the second child of this sum node...
...but that rules out $Z = 1! \quad \Rightarrow \quad P(Z = 2 \mid \mathbf{x}) = 1$
Likewise, if the first child is non-zero:

$$\Rightarrow P(Z = 1 | \mathbf{x}) = 1$$
Likewise, if the first child is non-zero:

\[ P(Z = 1 \mid x) = 1 \]

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
For each sum node, we know

1. if it is reached \((ctx = 1)\)
2. which child it selects
For each sum node, we know
1. if it is reached ($ctx = 1$)
2. which child it selects
For each sum node, we know
1. if it is reached ($ctx = 1$)
2. which child it selects

\[ \Rightarrow \quad \text{MLE by counting!} \]
Exact ML

Tractable MAR/MAP (smooth, decomposable, deterministic)

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

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

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 $D$, the maximum-likelihood sum-weights are:

$$w_{i,j}^{ML} = \frac{\sum_{x \in D} 1\{x \models [i \land j]\}}{\sum_{x \in D} 1\{x \models [i]\}} \leftarrow ct.x_i = 1, Z_i = j$$

---

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 $D$, the maximum-likelihood sum-weights are:

$$w_{i,j}^{ML} = \frac{\sum_{x \in D} 1\{x \models [i \land j]\}}{\sum_{x \in D} 1\{x \models [i]\}} \quad \leftarrow \text{ctx} x_i = 1, Z_i = j$$

$$\leftarrow \text{ctx} x_i = 1$$

---

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:

$$w_{i,j}^{\text{ML}} = \frac{\sum_{x \in \mathcal{D}} 1\{x = [i \land j]\}}{\sum_{x \in \mathcal{D}} 1\{x = [i]\}} \quad \leftarrow \; \text{ct} x_i = 1, Z_i = j$$

$$\leftarrow \; \text{ct} x_i = 1$$

$\Rightarrow$ 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;
data, structure = load(...);
um_examples(data) 17412
num_edges(structure) 270448
@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/
Formulate a prior $p(w, \theta)$ over sum-weights and leaf-parameters and perform posterior inference:

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

- 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

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deterministic</strong></td>
<td>?</td>
</tr>
<tr>
<td>closed-form MLE [Kisa et al. 2014b; Peharz et al. 2014]</td>
<td>?</td>
</tr>
<tr>
<td><strong>Non-deterministic</strong></td>
<td>?</td>
</tr>
<tr>
<td>EM [Poon et al. 2011; Peharz 2015; Zhao et al. 2016b]</td>
<td>?</td>
</tr>
<tr>
<td>SGD [Sharir et al. 2016; Peharz et al. 2019b]</td>
<td>?</td>
</tr>
<tr>
<td>Bayesian [Jaini et al. 2016; Rashwan et al. 2016]</td>
<td>?</td>
</tr>
<tr>
<td>[Zhao et al. 2016a; Trapp et al. 2019; Vergari et al. 2019]</td>
<td>?</td>
</tr>
<tr>
<td><strong>Discriminative</strong></td>
<td>?</td>
</tr>
</tbody>
</table>
Image-tailored (handcrafted) structures

“Recursive Image Slicing”

Image-tailored (handcrafted) structures

“Recursive Image Slicing”

Image-tailored (handcrafted) structures

“Recursive Image Slicing”

Image-tailored (handcrafted) structures

“Recursive Image Slicing”

Image-tailored (handcrafted) structures

“Recursive Image Slicing”

Image-tailored (handcrafted) structures

“Recursive Image Slicing”

Image-tailored (handcrafted) structures

“Recursive Image Slicing”

Image-tailored (handcrafted) structures

“Recursive Image Slicing”

Image-tailored (handcrafted) structures

“Recursive Image Slicing”

Image-tailored (handcrafted) structures

“Recursive Image Slicing”

Image-tailored (handcrafted) structures

“Recursive Image Slicing”

Image-tailored (handcrafted) structures

"Recursive Image Slicing"

⇒ Smooth & Decomposable

Image-tailored (handcrafted) structures

“Recursive Image Slicing”

⇒ Smooth & Decomposable
⇒ Tractable MAR

Learning the structure from data

“Recursive Data Slicing” — LearnSPN

Cluster

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

“Recursive Data Slicing” — LearnSPN

Cluster → **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
Try to find independent groups of random variables
Success → **product node**
LearnSPN: 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 → 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 → 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
Try to find independent groups of random variables
Fail → cluster → **sum node**
“Recursive Data Slicing” — LearnSPN

⇒ Continue until no further leaf can be expanded.
⇒ Clustering ratios also deliver (initial) parameters.
⇒ 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]
Structure Learning + MAP (determinism)

“Recursive conditioning” — Cutset Networks

[Rahman et al. 2014]

A B C D E F

Select Variable
Structure Learning + MAP (determinism)

“Recursive conditioning” — Cutset Networks

[Rahman et al. 2014]
"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]
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 B C D E F

Stop → learn Chow-Liu

A

0.55

E

0.3

0.7

B

0.35

0.65


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]
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]

...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...
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

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>deterministic</td>
<td>greedy</td>
</tr>
<tr>
<td>non-deterministic</td>
<td>[Rahman et al. 2014; Vergari et al. 2015]</td>
</tr>
<tr>
<td>Bayesian [Jaini et al. 2016; Rashwan et al. 2016]</td>
<td>[Dennis et al. 2015; Liang et al. 2017a; Dang et al. 2020]</td>
</tr>
</tbody>
</table>

### 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

- ?
- ?
### EVI inference: density estimation

<table>
<thead>
<tr>
<th>dataset</th>
<th>single models</th>
<th>ensembles</th>
<th>dataset</th>
<th>single models</th>
<th>ensembles</th>
</tr>
</thead>
<tbody>
<tr>
<td>jester</td>
<td>-52.42 [BNP-SPN]</td>
<td>-51.29 [LearnPSDDs]</td>
<td>webkb</td>
<td>-151.84 [ID-SPN]</td>
<td>-149.20 [XCNets]</td>
</tr>
<tr>
<td>netflix</td>
<td>-56.36 [ID-SPN]</td>
<td>-55.71 [LearnPSDDs]</td>
<td>cr52</td>
<td>-83.35 [ID-SPN]</td>
<td>-81.87 [XCNets]</td>
</tr>
</tbody>
</table>

123/159
## Learning probabilistic circuits

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Generative</strong></td>
<td><strong>Structure</strong></td>
</tr>
<tr>
<td><strong>deterministic</strong></td>
<td>greedy</td>
</tr>
<tr>
<td>closed-form MLE</td>
<td>top-down</td>
</tr>
<tr>
<td>[Kisa et al. 2014b; Peharz et al. 2014]</td>
<td>[Gens et al. 2013; Rooshenas et al. 2014]</td>
</tr>
<tr>
<td><strong>non-deterministic</strong></td>
<td>bottom-up</td>
</tr>
<tr>
<td>EM</td>
<td>[Rahman et al. 2014; Vergari et al. 2015]</td>
</tr>
<tr>
<td>[Poon et al. 2011; Peharz 2015; Zhao et al. 2016b]</td>
<td>[Peharz et al. 2013]</td>
</tr>
<tr>
<td>SGD</td>
<td>hill climbing</td>
</tr>
<tr>
<td>Bayesian</td>
<td>random</td>
</tr>
<tr>
<td><strong>Discriminative</strong></td>
<td></td>
</tr>
<tr>
<td><strong>deterministic</strong></td>
<td>greedy</td>
</tr>
<tr>
<td>convex-opt MLE</td>
<td>top-down</td>
</tr>
<tr>
<td>[Liang et al. 2019]</td>
<td>[Shao et al. 2019]</td>
</tr>
<tr>
<td><strong>non-deterministic</strong></td>
<td>hill climbing</td>
</tr>
<tr>
<td>EM</td>
<td>[Rooshenas et al. 2016; Liang et al. 2019]</td>
</tr>
<tr>
<td>[Rashwan et al. 2018]</td>
<td></td>
</tr>
<tr>
<td>SGD</td>
<td></td>
</tr>
<tr>
<td>[Gens et al. 2012; Sharir et al. 2016]</td>
<td></td>
</tr>
<tr>
<td>[Peharz et al. 2019b]</td>
<td></td>
</tr>
</tbody>
</table>
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:

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

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

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

$$(\mathbb{B} = \{0, 1\}, \lor, \land, 0, 1)$$

⇒ **logical circuits!**
Logical circuits are compact representations for boolean functions...

**Logical circuits**

- **s/d-D/NNFs**
  - [Darwiche et al. 2002a]

- **O/BDDs**
  - [Bryant 1986]

- **SDDs**
  - [Darwiche 2011]
Logical circuits

structural properties

...and like probabilistic 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
A task called **weighted model counting** (WMC)

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

- Probabilistic inference by WMC:
  1. Encode probabilistic model as WMC formula \( \Delta \)
  2. Compile \( \Delta \) 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
⇒ parameter variables → edge parameters

Compiled circuit of WMC encoding

Equivalent probabilistic circuit
From BN trees to circuits
via compilation
From BN trees to circuits via compilation

Bottom-up *compilation*: starting from leaves...
From BN trees to circuits via compilation

...compile a leaf CPT

\[ p(A|C = 0) \]

\[ A = 0 \quad A = 1 \]

\[ .3 \quad .7 \]
From BN trees to circuits
via compilation

...compile a leaf CPT

$$p(A|C = 1)$$

$$\begin{align*}
A = 0 & \quad .6 \\
A = 1 & \quad .4
\end{align*}$$
From BN trees to circuits via compilation

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

```
p(A|C)
p(B|C)
```

```
A = 0  A = 1  B = 0  B = 1
A = 0  A = 1  B = 0  B = 1
```
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

\[ p(D) \]

\[ D = 0 \]
\[ D = 1 \]

\[ C = 0 \]
\[ C = 1 \]

\[ A = 0 \]
\[ A = 1 \]
\[ B = 0 \]
\[ B = 1 \]
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
Vlasselaer et al., “Exploiting Local and Repeated Structure in Dynamic Bayesian Networks”, 2016
<table>
<thead>
<tr>
<th>Model</th>
<th>smooth</th>
<th>dec.</th>
<th>det.</th>
<th>str.dec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic Circuits (ACs)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>×</td>
</tr>
<tr>
<td>Sum-Product Networks (SPNs)</td>
<td>✔</td>
<td>✔</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Cutset Networks (CNets)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>×</td>
</tr>
<tr>
<td>Probabilistic Decision Graphs</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>PSDDs</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>AndOrGraphs</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
Structured decomposability

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

\[ X_3 \times X_1 \times X_2 \times X_3 \times X_1 \times X_2 \times X_3 \times X_1 \times X_2 \]

\[ \Rightarrow \]

**stronger requirement than decomposability**
Structured decomposability

A product node is structured decomposable if decomposes according to a node in a \( \text{vtree} \) \( \Rightarrow \) stronger requirement than decomposability

\[ X_1 \times X_2 \times X_3 \]

\text{vtree}

\text{non structured decomposable circuit}
Questions: What is the probability of having a traffic jam on my route to campus?
**Probability of logical events**

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

$q_8(m) = p_m(\bigvee_{i \in \text{route}} \text{JamStr}_i)$

$\Rightarrow$ marginal + logical events
**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 \( p(x) = \sum_i w_i p_i(x) \), \( \alpha = \vee_j \alpha_j \),

(smooth \( p \))

(smooth + deterministic \( \alpha \)):

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

\[\Rightarrow\] probabilities are “pushed down” to children

\[X_1 > 0.6 \quad \neg X_2 \quad X_1 \leq 0.3\]
Smoothness + structured decomp. = tractable PR

If \( p(x, y) = p(x)p(y) \), \( \alpha = \beta \land \gamma \),
(structured decomposability):

\[
p(\alpha) = p(\beta \land \gamma) \cdot p(\beta \land \gamma) = p(\beta) \cdot p(\gamma)
\]

\( \implies \text{probabilities decompose into simpler ones} \)
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
  $\Rightarrow$ cache the values!
- feedforward evaluation (bottom-up)
Smoothness + structured decomp. = tractable PR

To compute $p(\alpha)$:

- compute the probability for each pair of probabilistic and logical circuit nodes for the same vtree node
  $\Rightarrow$ cache the values!

- feedforward evaluation (bottom-up)
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 $f$ given a distribution $p$ over the input features

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

⇒ PR if $f$ is a logical formula
⇒ missing values at test time
⇒ exploratory classifier analysis

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

Q8: 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/
using ProbabilisticCircuits

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

Q9: Is the predictive model biased by gender?

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/
1. How precise is the characterization of tractable circuits by structural properties?  
   ⇒ necessary conditions

2. How do structural constraints affect the circuit sizes?  
   ⇒ succinctness analysis

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.

⇒ Are these properties necessary?

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

\[ \Rightarrow \text{decomposability implies consistency} \]

**Consistent circuit**

**Inconsistent circuit**
Determinism + consistency = tractable MAP
**Determinism** + **consistency** = **tractable MAP**

\[
\text{If } \max_{q_{\text{shared}}} p(q, e) = \\
\max_{q_{\text{shared}}} p(q_x, e_x) \cdot \max_{q_{\text{shared}}} p(q_y, e_y) \text{ (consistent):} \\
\max_q p(q, e) = \max_{q_x, q_y} p(q_x, e_x, q_y, e_y) \\
= \max_{q_x} p(q_x, e_x) \cdot \max_{q_y} p(q_y, e_y) \\
\implies \text{ solving optimization independently}
\]
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 $m_2 \in \mathcal{M}_2$, there exists $m_1 \in \mathcal{M}_1$ 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

: strictly more succinct
Expressive efficiency of circuits

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

[Peharz et al. 2015]*: strictly more succinct
[Darwiche et al. 2002b]: equally succinct

MAP

smooth & cons.
det. & Decomp.
smooth & Decomp.

MAR

: strictly more succinct
: equally succinct
Expressive efficiency of circuits

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

- [Peharz et al. 2015]*
- [Darwiche et al. 2002b]
Consider the following circuit over Boolean variables:
\[
\prod_i (Y_i \cdot Z_{i1} + (\neg Y_i) \cdot Z_{i2}), \quad Z_{ij} \in X
\]
- Size linear in the number of variables
- Deterministic and consistent
- Marginal (with no evidence) is the solution to \#P-hard SAT' problem [Valiant 1979] \(\implies\) no tractable circuit for marginals!
Expressive efficiency of circuits

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

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

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

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

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

- Linear-size smooth and decomposable circuit
- MAP of $p(X)$ solves marginal MAP of $p(X, C)$ which is NP-hard [de Campos 2011]

$\Rightarrow$ no tractable circuit for MAP!
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”
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

depth 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*  
github.com/arranger1044/awesome-spn


Darwiche, Adnan (2003). “A Differential Approach to Inference in Bayesian Networks”. In: J.ACM.


References IV


References VI


References VII


Friesen, Abram L and Pedro Domingos (2016). “Submodular Sum-product Networks for Scene Understanding”. In:


References IX


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)*.


References X


References XI


References XIII


