Probabilistic Circuits

Inference Representations

Learning Theory

Antonio Vergari
University of California, Los Angeles

Guy Van den Broeck
University of California, Los Angeles

Robert Peharz
TU Eindhoven

YooJung Choi
University of California, Los Angeles

May 12th, 2020 - CS201 - UCLA
The Alphabet Soup of probabilistic models
Fully factorized

NaiveBayes  AndOrGraphs  PDGs

Trees  PSDDs  CNets  LTM s  SPNs  NADES

Thin Junction Trees  ACs  MADEs  MAFs  VAEs

Polytrees  FVSBNs  TACs  IAFs  NAFs  RAEs

Mixtures  BNs  NICE  FGs  GANs

RealNVP  MNs

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
Today 12th May

Why tractable inference?
or expressiveness vs tractability

Probabilistic circuits
a unified framework for tractable probabilistic modeling

Thursday 14th May

Learning circuits
learning their structure and parameters from data
Today 12th May

**Why tractable inference?**

or expressiveness vs tractability

**Probabilistic circuits**

a unified framework for tractable probabilistic modeling

Thursday 14th May

**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?

$q_1$: 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?”

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!
\( p_m(X) \approx \ldots \) 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.?

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

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

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

$X = \{\text{Day, Time, Jam}_{\text{str1}}, \text{Jam}_{\text{str2}}, \ldots, \text{Jam}_{\text{strN}}\}$

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

$\Rightarrow$ marginals
Why probabilistic inference?

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

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

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

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

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

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

$\implies \text{marginals + MAP + logical events}$
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 $O(poly(|m|))$. 
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 $O(poly(|m|))$.

\[ \Rightarrow \text{often poly will in fact be linear!} \]
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 $Q$ 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 $\text{poly}$ 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 $O(\text{poly}(|X|))$.

$\Rightarrow$ 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.?*

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

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

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

- 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 et al., “Auto-Encoding Variational Bayes”, 2014
Variational Autoencoders

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

- an explicit likelihood model!
- ... but computing \( \log p_\theta(x) \) is intractable
  \( \Rightarrow \) an infinite and uncountable mixture
  \( \Rightarrow \) no tractable EVI
- we need to optimize the ELBO...
  \( \Rightarrow \) which is “tricky” [Alemi et al. 2017; Dai et al. 2019; Ghosh et al. 2019]
tractable bands
\[ p_X(x) = p_Z(f^{-1}(x)) \left| \det \left( \frac{\delta f^{-1}}{\delta x} \right) \right| \]

- an explicit likelihood!
- ...plus structured Jacobians \[ \Rightarrow \text{tractable EVI queries!} \]
- 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

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

- 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 \( \Rightarrow \text{tractable EVI queries!} \)
- 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.?

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

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

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

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

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

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

$$p_m(q | e) = \frac{p_m(q, e)}{p_m(e)}$$
**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*

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

⇒ tractable EVI queries!
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!
- **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

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

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

\[ \text{[Dagum et al. 1993; Roth 1996]} \]
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].

⇒ what about bounding the treewidth by design?
Low-treewidth PGMs

- **Trees**
  - ![Tree Diagram]
  - [Meilă et al. 2000]

- **Polytrees**
  - ![Polytree Diagram]
  - [Dasgupta 1999]

- **Thin Junction trees**
  - ![Thin Junction Tree Diagram]
  - [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 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 ...

Martens et al., “On the Expressive Efficiency of Sum Product Networks”, 2014
Mixtures as a convex combination of $k$ (simpler) probabilistic models

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

EVI, MAR, CON queries scale linearly in $k$
Mixtures as a convex combination of $k$ (simpler) probabilistic models

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

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

$\Rightarrow$ increased expressiveness
Expressiveness and efficiency

Expressiveness: Ability to represent rich and effective classes of functions

\[ \implies \text{mixture of Gaussians can approximate any distribution!} \]

Martens et al., “On the Expressive Efficiency of Sum Product Networks”, 2014
Expressiveness and efficiency

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

⇒ mixture of Gaussians can approximate any distribution!

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

⇒ but how many components does a Gaussian mixture need?

---

Martens et al., “On the Expressive Efficiency of Sum Product Networks”, 2014
How expressive efficient are mixture?
How expressive efficient are mixture?
How expressive efficient are mixture?
How expressive efficient are mixture?
How expressive efficient are mixture?
How expressive efficient are mixture?
How expressive efficient are mixture?
How expressive efficient are mixture?

stack mixtures like in deep generative models
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?

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

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

General: $\arg\max_q p_m(q \mid e)$

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

Q6: 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?

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

$\implies$ \text{NP$^{PP}$-complete} \cite{Park2006}

$\implies$ \text{NP-hard for trees} \cite{Campos2011}

$\implies$ \text{NP-hard even for Naive Bayes} \cite{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?

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

⇒ 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?

$\Rightarrow$ counts + group comparison

**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. 2019b]

---

_Bekker et al., “Tractable Learning for Complex Probability Queries”, 2015_
tractable bands
<table>
<thead>
<tr>
<th>$\mathcal{M}$</th>
<th>Q:</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>
<td></td>
</tr>
<tr>
<td>VAEs</td>
<td>✗</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>
<td></td>
</tr>
<tr>
<td>Trees</td>
<td>✗</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>
<td></td>
</tr>
<tr>
<td>?</td>
<td></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**!

\[ \Rightarrow \text{ but definitely not expressive...} \]
tractable bands
Expressive models are not very tractable...
and tractable ones are not very expressive...
probabilistic circuits are at the “sweet spot”
Probabilistic Circuits
Probabilistic circuits

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

A probabilistic circuit $C$ over variables $X$ is a computational graph encoding a (possibly unnormalized) probability distribution $p(X)$
Probabilistic circuits

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

$\implies$ operational semantics!

$\implies$ by constraining the graph we can make inference tractable...
1. What are the building blocks of probabilistic circuits? 
   \[ \Rightarrow \] How to build a tractable computational graph?

2. For which queries are probabilistic circuits tractable? 
   \[ \Rightarrow \] 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
Distributions as computational graphs

Base case: a single node encoding a distribution

⇒ e.g., indicators for $X$ or $\neg X$ for Boolean random variable
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 $Z$ (unnormalized)
- **MAP**: output the mode
Factorizations as product nodes

Divide and conquer complexity

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

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

Divide and conquer complexity

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

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

Divide and conquer complexity

\[ p(x_1, x_2, x_3) = p(x_1) \cdot p(x_2) \cdot p(x_3) \]
Factorizations as product nodes

Divide and conquer complexity

\[ p(x_1, x_2, x_3) = p(x_1) \cdot p(x_2) \cdot p(x_3) \]
Mixtures as sum nodes
Enhance expressiveness

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

\( \Rightarrow \) e.g. modeling a mixture of Gaussians...
Mixtures as sum nodes

Enhance expressiveness

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

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

Enhance expressiveness

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

⇒ 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
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>unit of computations</td>
</tr>
<tr>
<td><strong>Edges:</strong></td>
<td>order of execution</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>
<tr>
<td></td>
<td>feedforward pass</td>
</tr>
<tr>
<td></td>
<td>backward pass</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!

⇒ structural constraints needed for tractability
Which structural constraints to ensure tractability?
**Decomposability**

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

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

**decomposable circuit**

\[
\times \quad X_1 \quad X_1 \quad X_3
\]

**non-decomposable** circuit

---

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

*aka completeness*

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

$$\implies \text{otherwise not accounting for some variables}$$

**smooth circuit**

$$\begin{array}{c}
\begin{array}{c}
\land \\
X_1 \\
\end{array}
\end{array} + 
\begin{array}{c}
\land \\
X_1 \\
\end{array}$$

**non-smooth** circuit

$$\begin{array}{c}
\begin{array}{c}
\land \\
X_1 \\
\end{array} + 
\begin{array}{c}
\land \\
X_2 \\
\end{array}$$

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

Darwiche et al., “A knowledge compilation map”, 2002
**Smoothness** + **decomposability** = **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):

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

\[\Rightarrow \text{integrals decompose into easier ones}\]
\[ \text{Smoothness} + \text{decomposability} = \text{tractable MAR} \]

Forward pass evaluation for MAR

\[ \Rightarrow \text{linear in circuit size!} \]

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 \text{EVI}

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

Forward pass evaluation for MAR

\[ \Rightarrow \text{linear in circuit size!} \]

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

- Leaf over \( X_1 \) and \( X_3 \) output \( Z_i = \int p(x_i) dx_i \)
  \[ \Rightarrow \text{for normalized leaf distributions: } 1.0 \]

- Leaf over \( X_2 \) and \( X_4 \) output \( \text{EVI} \)

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

Forward pass evaluation for MAR

\[ \Rightarrow \quad \text{linear in circuit size!} \]

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

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

\[ \Rightarrow \quad \text{for normalized leaf distributions: } \mathbf{1.0} \]

-叶片 over \( X_2 \) and \( X_4 \) output **EVI**

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

Analogously, for arbitrary conditional queries:

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

1. evaluate \( p(q, e) \) \( \Rightarrow \) one feedforward pass
2. evaluate \( p(e) \) \( \Rightarrow \) another feedforward pass
   \( \Rightarrow \) ...still linear in circuit size!
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:

$$\arg\max_{q} p(q | e)$$
**Smoothness** + **decomposability** = **tractable MAP**

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

\[
\arg\max_q p(q | e)
\]

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

\[
\arg\max_q \sum_i w_i p_i(q, e) = \arg\max_q \sum_z p(q, z, e) \neq \sum_z \arg\max_q p(q, z, e)
\]

⇒ **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

\[ w_1 \times X_1 \leq \theta \times X_2 \]

\[ w_2 \times X_1 > \theta \times X_2 \]

\[ w_1 \times X_1 \times X_2 \]

\[ w_2 \times X_1 > \theta \times X_2 \]

\[ w_1 \times X_1 \times X_2 \]

\[ w_2 \times X_1 \times X_2 \]

\[ X_1 \leq \theta \]

\[ X_2 \times \]

\[ X_1 > \theta \]

\[ X_2 \times \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]
Determinism + decomposability = tractable MAP

Computing maximization with arbitrary evidence \( e \)

\[ \max_q p(q \mid e) \]

E.g., suppose we want to compute:

\[ \times \times \times \times \times \times X_1 \times X_2 X_1 X_2 X_3 X_4 X_3 X_4 \]

linear in circuit size!
Determinism + decomposability = tractable MAP

If \( p(q, e) = \sum_i w_ip_i(q, e) = \max_i w_ip_i(q, e) \),

(deterministic sum node):

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

\( \Rightarrow \) one non-zero child term, thus sum is max
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):

\[
\max_q p(q \mid e) = \max_q p(q, e) = \max_{q_{x},q_{y}} p(q_x, e_x, q_y, e_y) = \max_{q_{x}} p(q_x, e_x) \cdot \max_{q_{y}} p(q_y, e_y)
\]

\[\Rightarrow \text{solving optimization independently}\]
Determinism + decomposability = tractable MAP

Evaluating the circuit twice: 

*bottom-up* and *top-down* → still linear in circuit size!
\[\text{Determinism} + \text{decomposability} = \text{tractable MAP}\]

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 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**  \(\Rightarrow\)  \textit{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

![Diagram](image-url)
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
Friesen et al., “Submodular Sum-product Networks for Scene Understanding”, 2016
Determinism + decomposability = tractable MMAP

Analogously, we could also do a MMAP query:

$$\text{argmax}_{q} \sum_{z} p(q, z \mid e)$$
Determinism + decomposability = tractable MMAP

We cannot decompose a MMAP query!

$$\arg\max_q \sum_z p(q, z | e)$$

we still have latent variables to marginalize...

We need more structural properties!

⇒ more advanced queries tomorrow...
where are probabilistic circuits?
tractability vs expressive efficiency
Low-treewidth PGMs

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

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

ACs [Darwiche 2003] are decomposable, smooth, and deterministic. They support tractable EVI, MAR/CON, and MAP.

\[ \begin{align*}
\text{ACs} & \quad \text{are} \\
\text{decomposable} & \quad \text{They support tractable} \\
\text{smooth} & \quad \text{EVI} \\
\text{deterministic} & \quad \text{MAR/CON} \\
\text{parameters are attached to the leaves} & \quad \text{MAP} \\
\text{...but can be moved to the sum node edges } [\text{Rooshenas et al. 2014}] \\
\end{align*} \]

Lowd et al., “Learning Markov Networks With Arithmetic Circuits”, 2013
Sum-Product Networks (SPNs)

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
PSDDs [Kisa et al. 2014] 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
AndOrGraphs

AndOrGraphs [Dechter et al. 2007] are
- structured
- decomposable
- smooth
- deterministic

They support tractable
- EVI
- MAR/CON
- MAP
- Complex queries!

Dechter et al., “AND/OR search spaces for graphical models”, 2007
Marinescu et al., “Best-first AND/OR search for 0/1 integer programming”, 2007
tractability vs expressive efficiency
How expressive are probabilistic circuits?

Measuring average test set log-likelihood on 20 density estimation benchmarks

Comparing against intractable models:

- Bayesian networks (BN) [Chickering 2002] with sophisticated context-specific CPDs
- MADEs [Germain et al. 2015]
- VAEs [Kingma et al. 2014] (IWAE ELBO [Burda et al. 2015])

Gens et al., “Learning the Structure of Sum-Product Networks”, 2013
**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>-5.99</td>
<td>-6.02</td>
<td>-6.04</td>
<td>-5.99</td>
<td>dna</td>
<td>-79.88</td>
<td>-80.65</td>
<td>-82.77</td>
<td>-94.56</td>
</tr>
<tr>
<td>msnbc</td>
<td>-6.04</td>
<td>-6.04</td>
<td>-6.06</td>
<td>-6.09</td>
<td>kosarek</td>
<td>-10.52</td>
<td>-10.83</td>
<td></td>
<td>-10.64</td>
</tr>
<tr>
<td>plants</td>
<td>-11.84</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>-33.19</td>
</tr>
<tr>
<td>audio</td>
<td>-39.39</td>
<td>-40.50</td>
<td>-38.95</td>
<td>-38.67</td>
<td>movie</td>
<td>-50.34</td>
<td>-54.37</td>
<td>-48.7</td>
<td>-47.43</td>
</tr>
<tr>
<td>jester</td>
<td>-51.29</td>
<td>-51.07</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>-146.9</td>
</tr>
<tr>
<td>netflix</td>
<td>-55.71</td>
<td>-57.02</td>
<td>-55.16</td>
<td>-54.73</td>
<td>cr52</td>
<td>-81.87</td>
<td>-87.56</td>
<td>-82.80</td>
<td>-81.33</td>
</tr>
<tr>
<td>accidents</td>
<td>-26.89</td>
<td>-26.32</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>-146.9</td>
</tr>
<tr>
<td>retail</td>
<td>-10.72</td>
<td>-10.87</td>
<td>-10.81</td>
<td>-10.83</td>
<td>bbc</td>
<td>-229.21</td>
<td>-257.86</td>
<td>-242.40</td>
<td>-240.94</td>
</tr>
</tbody>
</table>
Hybrid intractable + tractable EVI

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

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

Tan et al., “Hierarchical Decompositional Mixtures of Variational Autoencoders“, 2019
Conclusions
Why tractable inference?

or expressiveness vs tractability

Probabilistic circuits

a unified framework for tractable probabilistic modeling
Today  12th May

**Why tractable inference?**

or expressiveness vs tractability

**Probabilistic circuits**

a unified framework for tractable probabilistic modeling

Thursday  14th May

**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
**Readings**

*Probabilistic circuits: Representation and Learning*
starai.cs.ucla.edu/papers/LecNoAAAI20.pdf

*Foundations of Sum-Product Networks for probabilistic modeling*
tinyurl.com/w65po5d
References I


References II


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


References IV


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


References VI


Probabilistic Circuits

Inference Representations

Learning Theory

Robert Peharz
TU Eindhoven

YooJung Choi
University of California, Los Angeles

Antonio Vergari
University of California, Los Angeles

Guy Van den Broeck
University of California, Los Angeles

May 14th, 2020 - CS201 - UCLA
Why tractable inference?

or expressiveness vs tractability

Probabilistic circuits

a unified framework for tractable probabilistic modeling
Tuesday 12th May

Why tractable inference?
or expressiveness vs tractability

Probabilistic circuits
a unified framework for tractable probabilistic modeling

Today 14th May

Learning circuits
learning their structure and parameters from data

Advanced representations
tracing the boundaries of tractability and connections to other formalisms
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 $C$ over variables $X$ is a computational graph encoding a (possibly unnormalized) probability distribution $p(X)$ parameterized by $\Omega$.

Learning a circuit $C$ from data $D$ 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>
<tr>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
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?
Learning probabilistic circuits

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

...*end of Learning section!*
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

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

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

As simple as tossing a coin

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

⇒ maximum likelihood (ML) estimation over data $D$

E.g. Bernoulli with parameter $\theta$

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

⇒ Laplace smoothing
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:

\[ p_{\mathcal{L}}(x) = h(x) \exp(T(x)^T \theta - A(\theta)) \]

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":

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

\[ \hat{\theta}_{ML} = \theta(\hat{\phi}_{ML}) \]
The simplest “real” PC: a sum node

Recall that sum nodes represent *mixture models*:

\[ p_S(x) = \sum_{k=1}^{K} w_k p_{L_k}(x) \]
The simplest “real” PC: a sum node

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.

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

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

ML if \( Z \) was observed:

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

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

\[ w_k^{new} = \frac{\sum_{x \in \mathcal{D}} p(Z = k \mid x)}{\lvert \mathcal{D} \rvert} \quad \phi_k^{new} = \frac{\sum_{x, z \in \mathcal{D}} p(Z = k \mid x)T(x)}{\sum_{z \in \mathcal{D}} p(Z = k \mid x)} \]
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

\[ P(Z = 1) \quad \text{X} \quad w_1 \quad w_2 \quad P(Z = 2) \]

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

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

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

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

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

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

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

\[ P(X \mid Z = 1, ctx = 1) \]
\[ P(X \mid Z = 2, ctx = 1) \]
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: $p(ctx, Z_S \mid x)$
### 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)

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

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

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

\[\Rightarrow \text{This also works with missing values in } x!\]

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

*Tractable MAR (smooth, decomposable)*

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

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)

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

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

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

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

Darwiche, “A Differential Approach to Inference in Bayesian Networks”, 2003
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 $Z = 1! \quad \Rightarrow P(Z = 2 \mid \mathbf{x}) = 1$
Likewise, if the first child is non-zero:
\[ P(Z = 1 \mid x) = 1 \]
Likewise, if the first child is non-zero:

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

Thus, the latent variables are \textbf{actually observed} in deterministic circuits!
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
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

$\Rightarrow$ MLE by counting!
Exact ML

Tractable MAR/MAP (smooth, decomposable, deterministic)

Given a complete dataset $D$, the maximum-likelihood sum-weights are:

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

---

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:

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

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 $\mathcal{D}$, the maximum-likelihood sum-weights are:

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

$$\leftarrow ct x_i = 1$$

---

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:

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

$$\quad \quad \quad \leftarrow c t x_i = 1$$

$\Rightarrow$ 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
Bayesian parameter learning

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. 2016b]
- 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>Generative</strong></td>
<td></td>
</tr>
<tr>
<td>deterministic</td>
<td>closed-form MLE [Kisa et al. 2014a; Peharz et al. 2014]</td>
</tr>
<tr>
<td>non-deterministic</td>
<td>EM [Poon et al. 2011; Peharz 2015; Zhao et al. 2016a]</td>
</tr>
<tr>
<td></td>
<td>SGD [Sharir et al. 2016; Peharz et al. 2019]</td>
</tr>
<tr>
<td></td>
<td>Bayesian [Jaini et al. 2016; Rashwan et al. 2016]</td>
</tr>
<tr>
<td></td>
<td>[Zhao et al. 2016b; Trapp et al. 2019; Vergari et al. 2019]</td>
</tr>
<tr>
<td><strong>Discriminative</strong></td>
<td>?</td>
</tr>
<tr>
<td></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 et al., “Learning the Structure of Sum-Product Networks”, 2013
Learning the structure from data

“Recursive Data Slicing” — LearnSPN

Cluster $\rightarrow$ sum node

Gens et al., “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 et al., “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 et al., “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 et al., “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
Learning the structure from data

“Recursive Data Slicing” — LearnSPN

Single variable

Gens et al., “Learning the Structure of Sum-Product Networks”, 2013
Learning the structure from data

“Recursive Data Slicing” — LearnSPN

Single variable → leaf
Try to find independent groups of random variables

Gens et al., “Learning the Structure of Sum-Product Networks”, 2013
Try to find independent groups of random variables
Fail → cluster → $\text*{sum node}$
Continue until no further leaf can be expanded.

Clustering ratios also deliver (initial) parameters.
Learning the structure from data

“Recursive Data Slicing” — LearnSPN

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

Gens et al., “Learning the Structure of Sum-Product Networks”, 2013
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]
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]

\[ A \quad B \quad C \quad D \quad E \quad F \]

Split states

[Diagram showing a tree structure with nodes A, E, and B, and probabilities 0.55, 0.35, 0.45, 0.65, 0.3, 0.7]
"Recursive conditioning" — Cutset Networks

[**Structure Learning + MAP (determinism)**]

\[ A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F \]

Stop → learn Chow-Liu

![Diagram of a tree structure with edges and probabilities:]

- Edge A to E with probability 0.55
- Edge E to B with probability 0.45
- Edge B to E with probability 0.35
- Edge E to D with probability 0.7
- Edge D to E with probability 0.65

[**Rahman et al. 2014**]
Structure Learning + MAP (determinism)

“Recursive conditioning” — Cutset Networks

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

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. 2015a]
- Boosted CNets [Rahman et al. 2016]
- Randomized CNets, Bagging [Di Mauro et al. 2017]
Structure learning + MAP (determinism)

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

Structure learning as discrete optimization

Typical objective:

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

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

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

**Extremely Randomized CNets** (XCNets) [Di Mauro et al. 2017]
- Top-down random conditioning.
- Learning Chow-Liu trees at the leaves.
- Smooth, decomposable, deterministic.

**Random Tensorized SPNs** (RAT-SPNs) [Peharz et al. 2019]
- Random tree-shaped PCs.
- Discriminative+generative parameter learning (SGD/EM + dropout).
- Smooth, decomposable.
Ensembles of probabilistic circuits

Single circuits might be not accurate enough or overfit training data...
Solution: ensembles of circuits!

non-deterministic mixture models: another sum node!

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

Ensemble weights and components can be learned separately or jointly

- EM or structural EM
- bagging
- boosting
Bagging

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

Adding random subspace projection to bagged networks (like for CNets)
- more efficient than bagging

Di Mauro et al., “Learning Accurate Cutset Networks by Exploiting Decomposability”, 2015
Di Mauro et al., “Learning Bayesian Random Cutset Forests”, 2015
Boosting Probabilistic Circuits

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

\[
f_m = (1 - \eta_m)f_{m-1} + \eta_mc_m
\]

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

*Rahman et al., “Learning Ensembles of Cutset Networks”, 2016*
# Learning probabilistic circuits

## Parameters

<table>
<thead>
<tr>
<th>Generative</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{deterministic}</td>
<td>\textit{greedy}</td>
</tr>
<tr>
<td>\textit{non-deterministic}</td>
<td>\textit{hill climbing} [Lowd et al. 2008, 2013; Peharz et al. 2014]</td>
</tr>
<tr>
<td>[Zhao et al. 2016b; Trapp et al. 2019; Vergari et al. 2019]</td>
<td></td>
</tr>
</tbody>
</table>

## Discriminative

? ?

Note: The table layout is not perfect in the text, but it is clear in the image.
<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>
# Learning probabilistic circuits

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Generative</strong></td>
<td><strong>Greedy</strong></td>
</tr>
<tr>
<td><em>deterministic</em></td>
<td><strong>Top-down</strong></td>
</tr>
<tr>
<td>closed-form MLE</td>
<td>[Kisa et al. 2014a; Peharz et al. 2014]</td>
</tr>
<tr>
<td><em>non-deterministic</em></td>
<td><strong>Bottom-up</strong></td>
</tr>
<tr>
<td>EM</td>
<td>[Poon et al. 2011; Peharz 2015; Zhao et al. 2016a]</td>
</tr>
<tr>
<td>SGD</td>
<td></td>
</tr>
<tr>
<td>Bayesian</td>
<td><strong>Hill climbing</strong></td>
</tr>
<tr>
<td></td>
<td>[Jaini et al. 2016; Rashwan et al. 2016]</td>
</tr>
<tr>
<td></td>
<td>[Zhao et al. 2016b; Trapp et al. 2019; Vergari et al. 2019]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Discriminative</strong></th>
<th><strong>Greedy</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>deterministic</em></td>
<td><strong>Top-down</strong></td>
</tr>
<tr>
<td>convex-opt MLE</td>
<td>[Liang et al. 2019]</td>
</tr>
<tr>
<td><em>non-deterministic</em></td>
<td></td>
</tr>
<tr>
<td>EM</td>
<td>[Rashwan et al. 2018]</td>
</tr>
<tr>
<td>SGD</td>
<td>[Gens et al. 2012; Sharir et al. 2016]</td>
</tr>
<tr>
<td></td>
<td>[Peharz et al. 2019]</td>
</tr>
</tbody>
</table>

| Random              | **RAT-SPNs** [Peharz et al. 2019] **XCNet** [Di Mauro et al. 2017] |
Advanced Representations
From Part 1: **probabilistic circuits** unify tractable probabilistic models
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:

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

\[\implies \text{Algebraic model counting [Kimmig et al. 2017], Semi-ring programming [Belle et al. 2016]}\]

Historically, *very well studied for boolean functions*:

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

\[\implies \text{logical circuits!}\]
Logical circuits are compact representations for boolean functions...

$s/d$-D/NNFs

[Darwiche et al. 2002a]

$O/BDD$s

[Bryant 1986]

$SDD$s

[Darwiche 2011]
...and like probabilistic circuits, one can define **structural properties**: (structured) decomposability, smoothness, determinism allowing for tractable computations.

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

a knowledge compilation map

...inducing a hierarchy of tractable logical circuit families

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

connection to probabilistic circuits through WMC

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

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

- 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

\[ \text{parameter variables} \rightarrow \text{edge parameters} \]

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

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

\[ A = 0 \quad A = 1 \]
From BN trees to circuits via compilation

...compile a leaf CPT

\[ p(A|C = 1) \]

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

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

\[ p(A|C) \]
\[ p(B|C) \]

\[
\begin{align*}
A &= 0 & A &= 1 \\
B &= 0 & B &= 1
\end{align*}
\]
From BN trees to circuits via compilation

...and recurse over parents...

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

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

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

\[
D \quad C \quad A \quad B
\]
From BN trees to circuits
via compilation

...while reusing previously compiled nodes!...
From BN trees to circuits
via compilation

\[
p(D) = 0.5 \times 0.5 \times 0.5 = 0.125
\]
Compilation: probabilistic programming

```java
x = flip(θ₁);
if (x) {
  y = flip(θ₂)
} else {
  y = x
}
```

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></th>
<th>smooth</th>
<th>dec.</th>
<th>det.</th>
<th>str.decl.</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 (C Nets)</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 \textit{vtree} with a stronger requirement than decomposability.

\begin{itemize}
  \item \textit{vtree}
  \item \textit{structured decomposable circuit}
\end{itemize}
Structured decomposability

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

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

\[ \implies \] stronger requirement than decomposability

\[ \begin{array}{c}
\text{vtree} \\
X_3 \\
X_1 \quad X_2
\end{array} \]

\[ \begin{array}{c}
\text{non structured decomposable circuit} \\
X_3 \\
X_1 \quad X_3 \quad X_1 \quad X_3 \\
X_1 \quad X_2 \quad X_1 \quad X_2
\end{array} \]
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?

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

$\rightarrow$ **marginals + 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 \( p(x) = \sum_i w_i p_i(x) \), \( \alpha = \bigvee_j \alpha_j \),

**(smooth \( p \))

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

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

\[\Rightarrow \text{probabilities are “pushed down” to children}\]
Smoothness + structured decomp. = tractable PR

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

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

\( \Rightarrow \) probabilities decompose into simpler ones
\textbf{Smoothness} + \textbf{structured decomp.} = \textbf{tractable PR}

To compute $p(\alpha)$:

- compute the probability for each \textbf{pair} of probabilistic and logical circuit nodes for the \textbf{same vtree node}
  \[ \implies \text{cache the values!} \]

- feedforward evaluation (bottom-up)
To compute $p(\alpha)$:

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

  \[ X_1 > 0:6 : X_2 X_1  0:3 \]

  \[ X_3 \wedge X_2 \wedge X_1 \wedge X_3 \]

- feedforward evaluation (bottom-up)
structured decomposability $= \text{tractable...}$

- **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. 2015]
- **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

$\text{missing values at test time}$

$\text{exploratory classifier analysis}$

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

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

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

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

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?
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.

\[ X_1 X_2 X_3 \]

\[ w_1 w_2 w_3 w_4 \]

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

**Consistency**

**consistent circuit**

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

\[
\text{If } \max_{q_{\text{shared}}} p(q, e) = \\
\max_{q_{\text{shared}}} p(q_x, e_x) \cdot \max_{q_{\text{shared}}} p(q_y, e_y) \quad \text{(consistent)}:
\]

\[
\max_{q} p(q, e) = \max_{q_x, q_y} p(q_x, e_x, q_y, e_y) \\
= \max_{q_x} p(q_x, e_x) \cdot \max_{q_y} p(q_y, e_y)
\]

\[
\implies \text{solving optimization independently}
\]
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 $m_2 \in \mathcal{M}_2$, 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 $\mathcal{M}_1 \nleq \mathcal{M}_2$
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.

\[\text{Darwiche et al. 2002b}\]
Expressive efficiency of circuits

Smooth & decomposable circuits strictly more succinct than deterministic & decomposable ones

Smooth & consistent circuits are equally succinct as smooth & decomposable ones

- : strictly more succinct
--- : equally succinct
Smooth & decomposable circuits strictly more succinct than deterministic & decomposable ones.

Smooth & consistent circuits are equally succinct as smooth & decomposable ones.
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 [Valiant 1979] ⇒ no tractable circuit for marginals!
Consider following circuit over Boolean variables:
\[
\prod_i^{r} (Y_i \cdot Z_{i1} + (\neg Y_i) \cdot Z_{i2}), \quad Z_{ij} \in X
\]

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

⇒ "Complete the map"

More theoretical questions remaining

---

: strictly more succinct

: equally succinct
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: you can learn probabilistic circuits from data...
takeaway #2: or compile them from your favorite PGMs...
“What is the probability of having a traffic jam on my route to campus?”

\[ q(m) = p_m(\bigvee_{i \in \text{route}} \text{JamStr}_i) \]

takeaway #3: advanced structural properties enable advanced
probabilistic inference!
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/CS201.pdf
**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
References


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


References II


References IV


References VI


References VII


- 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)*.