Tractable Probabilistic Circuits

Guy Van den Broeck

VinAI Research - May 27, 2022
Outline

1. What are tractable probabilistic circuits?
2. Are these models any good?
3. How far can we push tractable inference?
4. What is their expressive power?
Outline

1. What are tractable probabilistic circuits?
2. Are these models any good?
3. How far can we push tractable inference?
4. What is their expressive power?
Intractable and tractable models
a unifying framework for tractable models
Probabilistic circuits

computational graphs that recursively define distributions

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
Probabilistic circuits

*computational graphs* that recursively define distributions

\[ p(X_1) = w_1 p_1(X_1) + w_2 p_2(X_1) \]

⇒

mixtures

\[ p(X) = p(Z = 1) \cdot p_1(X|Z = 1) + p(Z = 2) \cdot p_2(X|Z = 2) \]
Probabilistic circuits

computational graphs that recursively define distributions

\[
p(X_1) = w_1 p_1(X_1) + w_2 p_2(X_1)
\]
⇒ mixtures

\[
p(X_1, X_2) = p(X_1) \cdot p(X_2)
\]
⇒ factorizations
Likelihood \[ p(X_1 = -1.85, X_2 = 0.5, X_3 = -1.3, X_4 = 0.2) \]
Likelihood \[ p(X_1 = -1.85, X_2 = 0.5, X_3 = -1.3, X_4 = 0.2) \]
Likelihood

\[ p(X_1 = -1.85, X_2 = 0.5, X_3 = -1.3, X_4 = 0.2) \]
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
A sum node is \textit{smooth} if its children depend on the same set of variables.

A product node is \textit{decomposable} if its children depend on disjoint sets of variables.

\textit{smooth circuit}

\textit{decomposable circuit}

Darwiche and Marquis, “A Knowledge Compilation Map”, 2002
\textbf{Smoothness} + \textbf{decomposability} = \textbf{tractable MAR}

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

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

\implies \text{integrals are “pushed down” to children}
Smoothness + decomposability = tractable MAR

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

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

\( \Rightarrow \) integrals decompose into easier ones
Smoothness + decomposability = tractable MAR

Forward pass evaluation for MAR
\[ \implies \text{linear in circuit size!} \]

E.g. to compute \( p(x_2, x_4) \):
- Leaf nodes over \( X_1 \) and \( X_3 \) output \( Z_i = \int p(x_i) \, dx_i \)
  \[ \implies \text{for normalized leaf distributions: 1.0} \]
- Leaf nodes over \( X_2 \) and \( X_4 \) output \( \text{EVI} \)

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

Forward pass evaluation for MAR

$\Rightarrow$ linear in circuit size!

E.g. to compute $p(x_2, x_4)$:
- leafs over $X_1$ and $X_3$ output $Z_i = \int p(x_i) \, dx_i$
  $\Rightarrow$ for normalized leaf distributions: 1.0
- leafs over $X_2$ and $X_4$ output EVI
- feedforward evaluation (bottom-up)
Tractable MAR on PCs (Einsum Networks)

EVI 10,958.72 nats

MAR 5,387.55 nats

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

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

MAP for latent variable models is intractable [Conaty et al. 2017]
Outline

1. What are tractable probabilistic circuits?
2. Are these models any good?
3. How far can we push tractable inference?
4. What is their expressive power?
Learning Expressive Probabilistic Circuits

Hidden Chow-Liu Trees

Learned CLT structure captures strong pairwise dependencies

Learning Expressive Probabilistic Circuits

Hidden Chow-Liu Trees

Learned CLT structure captures strong pairwise dependencies

Learned HCLT structure

Compile into an equivalent PC

From BN trees to circuits via compilation
Learning Expressive Probabilistic Circuits

Hidden Chow-Liu Trees

Learned CLT structure captures strong pairwise dependencies

Learned HCLT structure

Compile into an equivalent PC

Mini-batch Stochastic Expectation Maximization

Lossless Data Compression

Encode

Data

10110

Bitstream

Decode

Reconstructed data

Expressive probabilistic model $p(x)$

+ Efficient coding algorithm

Determines the theoretical limit of compression rate

How close we can approach the theoretical limit

We want to compress a set of variables (e.g., pixels, letters) \( \{x_1, x_2, \ldots, x_k\} \)

Need to compute

- \( p(X_1 < x_1) \)
- \( p(X_1 \leq x_1) \)
- \( p(X_2 < x_2 | x_1) \)
- \( p(X_2 \leq x_2 | x_1) \)
- \( p(X_3 < x_3 | x_1, x_2) \)
- \( p(X_3 \leq x_3 | x_1, x_2) \)

\[ \vdots \]
Lossless Neural Compression with Probabilistic Circuits

Probabilistic Circuits
- Expressive → SoTA likelihood on MNIST.
- Fast → Time complexity of en/decoding is $O(|p| \log(D))$, where $D$ is the # variables and $|p|$ is the size of the PC.

Arithmetic Coding:

$p(X_1 < x_1)$
$p(X_1 \leq x_1)$
$p(X_2 < x_2 | x_1)$
$p(X_2 \leq x_2 | x_1)$
$p(X_3 < x_3 | x_1, x_2)$
$p(X_3 \leq x_3 | x_1, x_2)$
\[ \vdots \]
Lossless Neural Compression with Probabilistic Circuits

SoTA compression rates

<table>
<thead>
<tr>
<th>Dataset</th>
<th>HCLT (ours)</th>
<th>IDF</th>
<th>BitSwap</th>
<th>BB-ANS</th>
<th>JPEG2000</th>
<th>WebP</th>
<th>McBits</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>1.24 (1.20)</td>
<td>1.96 (1.90)</td>
<td>1.31 (1.27)</td>
<td>1.42 (1.39)</td>
<td>3.37</td>
<td>2.09 (1.98)</td>
<td></td>
</tr>
<tr>
<td>FashionMNIST</td>
<td>3.37 (3.34)</td>
<td>3.50 (3.47)</td>
<td>3.35 (3.28)</td>
<td>3.69 (3.66)</td>
<td>3.93</td>
<td>4.62 (3.72)</td>
<td></td>
</tr>
<tr>
<td>EMNIST (Letter)</td>
<td>1.84 (1.80)</td>
<td>2.02 (1.95)</td>
<td>1.90 (1.84)</td>
<td>2.29 (2.26)</td>
<td>3.62</td>
<td>3.31 (3.12)</td>
<td></td>
</tr>
<tr>
<td>EMNIST (ByClass)</td>
<td>1.89 (1.85)</td>
<td>2.04 (1.98)</td>
<td>1.91 (1.87)</td>
<td>2.24 (2.23)</td>
<td>3.61</td>
<td>3.34 (3.14)</td>
<td></td>
</tr>
</tbody>
</table>

Compress and decompress 5-40x faster than NN methods with similar bitrates

<table>
<thead>
<tr>
<th>Method</th>
<th># parameters</th>
<th>Theoretical bpd</th>
<th>Codeword bpd</th>
<th>Comp. time (s)</th>
<th>Decomp. time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC (HCLT, $M = 16$)</td>
<td>3.3M</td>
<td>1.26</td>
<td>1.30</td>
<td>9</td>
<td>44</td>
</tr>
<tr>
<td>PC (HCLT, $M = 24$)</td>
<td>5.1M</td>
<td>1.22</td>
<td>1.26</td>
<td>15</td>
<td>86</td>
</tr>
<tr>
<td>PC (HCLT, $M = 32$)</td>
<td>7.0M</td>
<td>1.20</td>
<td>1.24</td>
<td>26</td>
<td>142</td>
</tr>
<tr>
<td>IDF</td>
<td>24.1M</td>
<td>1.90</td>
<td>1.96</td>
<td>288</td>
<td>592</td>
</tr>
<tr>
<td>BitSwap</td>
<td>2.8M</td>
<td>1.27</td>
<td>1.31</td>
<td>578</td>
<td>326</td>
</tr>
</tbody>
</table>
Lossless Neural Compression with Probabilistic Circuits

Can be effectively combined with Flow models to achieve better generative performance

<table>
<thead>
<tr>
<th>Model</th>
<th>CIFAR10</th>
<th>ImageNet32</th>
<th>ImageNet64</th>
</tr>
</thead>
<tbody>
<tr>
<td>RealNVP</td>
<td>3.49</td>
<td>4.28</td>
<td>3.98</td>
</tr>
<tr>
<td>Glow</td>
<td>3.35</td>
<td>4.09</td>
<td>3.81</td>
</tr>
<tr>
<td>IDF</td>
<td>3.32</td>
<td>4.15</td>
<td>3.90</td>
</tr>
<tr>
<td>IDF++</td>
<td><strong>3.24</strong></td>
<td>4.10</td>
<td>3.81</td>
</tr>
<tr>
<td>PC+IDF</td>
<td>3.28</td>
<td><strong>3.99</strong></td>
<td><strong>3.71</strong></td>
</tr>
</tbody>
</table>
Tractable and expressive generative models of genetic variation data
PC Learners keep getting better! ... stay tuned ...

Table 1: Density estimation performance on MNIST-family datasets in test set bpd.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Sparse PC (ours)</th>
<th>HCLT</th>
<th>RatSPN</th>
<th>IDF</th>
<th>BitSwap</th>
<th>BB-ANS</th>
<th>McBits</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>1.14</td>
<td>1.20</td>
<td>1.67</td>
<td>1.90</td>
<td>1.27</td>
<td>1.39</td>
<td>1.98</td>
</tr>
<tr>
<td>EMNIST (MNIST)</td>
<td>1.52</td>
<td>1.77</td>
<td>2.56</td>
<td>2.07</td>
<td>1.88</td>
<td>2.04</td>
<td>2.19</td>
</tr>
<tr>
<td>EMNIST (Letters)</td>
<td>1.58</td>
<td>1.80</td>
<td>2.73</td>
<td>1.95</td>
<td>1.84</td>
<td>2.26</td>
<td>3.12</td>
</tr>
<tr>
<td>EMNIST (Balanced)</td>
<td>1.60</td>
<td>1.82</td>
<td>2.78</td>
<td>2.15</td>
<td>1.96</td>
<td>2.23</td>
<td>2.88</td>
</tr>
<tr>
<td>EMNIST (ByClass)</td>
<td>1.54</td>
<td>1.85</td>
<td>2.72</td>
<td>1.98</td>
<td>1.87</td>
<td>2.23</td>
<td>3.14</td>
</tr>
<tr>
<td>FashionMNIST</td>
<td>3.27</td>
<td>3.34</td>
<td>4.29</td>
<td>3.47</td>
<td>3.28</td>
<td>3.66</td>
<td>3.72</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PC</th>
<th>Bipartite flow</th>
<th>AF/SCF</th>
<th>IAF/SCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Penn Treebank</td>
<td>1.23</td>
<td>1.38</td>
<td>1.46</td>
<td>1.63</td>
</tr>
</tbody>
</table>
Training SotA likelihood full MNIST probabilistic circuit model in ~7 minutes on GPU:
https://github.com/Juice-jl/ProbabilisticCircuits.jl/blob/master/examples/train_mnist_hclt.ipynb

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PC (ours)</th>
<th>IDF</th>
<th>Hierarchical VAE</th>
<th>PixelVAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>1.20</td>
<td>2.90</td>
<td>1.27</td>
<td>1.39</td>
</tr>
<tr>
<td>FashionMNIST</td>
<td>3.34</td>
<td>3.47</td>
<td>3.20</td>
<td>3.65</td>
</tr>
<tr>
<td>EMNIST (Letter split)</td>
<td>1.80</td>
<td>1.95</td>
<td>1.84</td>
<td>2.28</td>
</tr>
<tr>
<td>EMNIST (ByClass split)</td>
<td>1.95</td>
<td>1.98</td>
<td>1.97</td>
<td>2.23</td>
</tr>
</tbody>
</table>

* Note: all reported numbers are bits-per-dimension (bpd). The results are extracted from [1].


We start by importing ProbabilisticCircuits.jl and other required packages:

```julia
using ProbabilisticCircuits
using MLDatasets
using CUDA
```

We first load the MNIST dataset from MLDatasets.jl and move them to GPU:
Expressive models without compromises
Outline

1. What are tractable probabilistic circuits?
2. Are these models any good?
3. How far can we push tractable inference?
4. What is their expressive power?
**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

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

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

A sum node is **deterministic** if only one of its children outputs non-zero for any input

\[ \Rightarrow \text{allows tractable MAP inference} \]

\[ \arg\max_x p(x) \]
If \( p(q, e) = \sum_i w_i p_i(q, e) = \max_i w_i p_i(q, e) \),

**deterministic** sum node:

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

\[\Rightarrow \text{one non-zero child term, thus sum is max}\]
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):

$$\max_q p(q | e) = \max_q p(q, e)$$

$$= \max_{q_x, q_y} p(q_x, e_x, q_y, e_y)$$

$$= \max_{q_x} p(q_x, e_x) \cdot \max_{q_y} p(q_y, e_y)$$

$\Rightarrow$ solving optimization independently
Queries as pipelines: KLD

\[ \text{KLD}(p \parallel q) = \int p(x) \times \log((p(x)/q(x))) dX \]
Queries as pipelines: Cross Entropy

\[ H(p, q) = \int p(x) \times \log(q(x)) \, dx \]

\[ \Rightarrow \text{we can reuse the operations!} \]
<table>
<thead>
<tr>
<th>Operation</th>
<th>Input conditions</th>
<th>Output conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOG</td>
<td>$\log(p)$</td>
<td>$\text{Sm, Dec, Det}$</td>
</tr>
</tbody>
</table>

smooth, decomposable, deterministic

smooth, decomposable
# Tractable circuit operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Input properties</th>
<th>Tractability</th>
<th>Hardness</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUM $\theta_1 p + \theta_2 q$</td>
<td>(+Cmp)</td>
<td>(+SD)</td>
<td>NP-hard for Det output</td>
</tr>
<tr>
<td>PRODUCT $p \cdot q$</td>
<td>Cmp (+Det, +SD)</td>
<td>Dec (+Det, +SD)</td>
<td>#P-hard w/o Cmp</td>
</tr>
<tr>
<td>POWER $p^n$, $n \in \mathbb{N}$</td>
<td>SD (+Det)</td>
<td>SD (+Det)</td>
<td>#P-hard w/o SD</td>
</tr>
<tr>
<td>POWER $p^\alpha$, $\alpha \in \mathbb{R}$</td>
<td>Sm, Dec, Det (+SD)</td>
<td>Sm, Dec, Det (+SD)</td>
<td>#P-hard w/o Det</td>
</tr>
<tr>
<td>QUOTIENT $p/q$</td>
<td>Cmp; $q$ Det (+p Det, +SD)</td>
<td>Dec (+Det, +SD)</td>
<td>#P-hard w/o Det</td>
</tr>
<tr>
<td>LOG $\log(p)$</td>
<td>Sm, Dec, Det</td>
<td>Sm, Dec</td>
<td>#P-hard w/o Det</td>
</tr>
<tr>
<td>EXP $\exp(p)$</td>
<td>linear</td>
<td>SD</td>
<td>#P-hard</td>
</tr>
</tbody>
</table>
Inference by tractable operations

systematically derive tractable inference algorithm of complex queries

<table>
<thead>
<tr>
<th>Query</th>
<th>Tract. Conditions</th>
<th>Hardness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross Entropy</td>
<td>Cmp, q Det</td>
<td>#P-hard w/o Det</td>
</tr>
<tr>
<td>Shannon Entropy</td>
<td>Sm, Dec, Det</td>
<td>coNP-hard w/o Det</td>
</tr>
<tr>
<td>Rényi Entropy</td>
<td>SD</td>
<td>#P-hard w/o SD</td>
</tr>
<tr>
<td>Mutual Information</td>
<td>Sm, SD, Det*</td>
<td>coNP-hard w/o SD</td>
</tr>
<tr>
<td>Kullback-Leibler Div.</td>
<td>Cmp, Det</td>
<td>#P-hard w/o Det</td>
</tr>
<tr>
<td>Rényi’s Alpha Div.</td>
<td>Cmp, q Det</td>
<td>#P-hard w/o Det</td>
</tr>
<tr>
<td>Itakura-Saito Div.</td>
<td>Cmp, Det</td>
<td>#P-hard w/o Det</td>
</tr>
<tr>
<td>Cauchy-Schwarz Div.</td>
<td>Cmp</td>
<td>#P-hard w/o Cmp</td>
</tr>
<tr>
<td>Squared Loss</td>
<td>Cmp</td>
<td>#P-hard w/o Cmp</td>
</tr>
</tbody>
</table>
Even harder queries

**Marginal MAP**

Given a set of query variables \( Q \subseteq X \) and evidence \( e \), find: \( \arg \max_q p(q|e) \)

\[ \Rightarrow \text{i.e. MAP of a marginal distribution on } Q \]

\( \text{NP}^{PP} \)-complete for PGMs

\( \text{NP-hard} \) even for PCs tractable for marginals, MAP & entropy
Pruning circuits

Any parts of circuit not relevant for MMAP state can be pruned away

e.g. $p(X_1 = 1, X_2 = 0)$

We can find such edges in *linear time*
Iterative MMAP solver

Prune edges

Tighten bounds

<table>
<thead>
<tr>
<th>Dataset</th>
<th>runtime (# solved)</th>
<th>search</th>
<th>pruning</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLTCS</td>
<td>0.01 (10)</td>
<td>0.63 (10)</td>
<td></td>
</tr>
<tr>
<td>MSNBC</td>
<td>0.03 (10)</td>
<td>0.73 (10)</td>
<td></td>
</tr>
<tr>
<td>KDD</td>
<td>0.04 (10)</td>
<td>0.68 (10)</td>
<td></td>
</tr>
<tr>
<td>Plants</td>
<td>2.95 (10)</td>
<td>2.72 (10)</td>
<td></td>
</tr>
<tr>
<td>Audio</td>
<td>2041.33 (6)</td>
<td>13.70 (10)</td>
<td></td>
</tr>
<tr>
<td>Jester</td>
<td>2913.04 (2)</td>
<td>14.74 (10)</td>
<td></td>
</tr>
<tr>
<td>Netflix</td>
<td>- (0)</td>
<td>47.18 (10)</td>
<td></td>
</tr>
<tr>
<td>Accidents</td>
<td>109.56 (10)</td>
<td>15.86 (10)</td>
<td></td>
</tr>
<tr>
<td>Retail</td>
<td>0.06 (10)</td>
<td>0.81 (10)</td>
<td></td>
</tr>
<tr>
<td>Pumsb-star</td>
<td>2208.27 (7)</td>
<td>20.88 (10)</td>
<td></td>
</tr>
<tr>
<td>DNA</td>
<td>- (0)</td>
<td>505.75 (9)</td>
<td></td>
</tr>
<tr>
<td>Kosarek</td>
<td>48.74 (10)</td>
<td>3.41 (10)</td>
<td></td>
</tr>
<tr>
<td>MSWeb</td>
<td>1543.49 (10)</td>
<td>1.28 (10)</td>
<td></td>
</tr>
<tr>
<td>Book</td>
<td>- (0)</td>
<td>46.50 (10)</td>
<td></td>
</tr>
<tr>
<td>EachMovie</td>
<td>- (0)</td>
<td>1216.89 (8)</td>
<td></td>
</tr>
<tr>
<td>WebKB</td>
<td>- (0)</td>
<td>575.68 (10)</td>
<td></td>
</tr>
<tr>
<td>Reuters-52</td>
<td>- (0)</td>
<td>120.58 (10)</td>
<td></td>
</tr>
<tr>
<td>20 NewsGrp.</td>
<td>- (0)</td>
<td>504.52 (9)</td>
<td></td>
</tr>
<tr>
<td>BBC</td>
<td>- (0)</td>
<td>2757.18 (3)</td>
<td></td>
</tr>
<tr>
<td>Ad</td>
<td>- (0)</td>
<td>1254.37 (8)</td>
<td></td>
</tr>
</tbody>
</table>

Probabilistic Sufficient Explanations

**Goal**: explain an instance of classification (a specific prediction)

Explanation is a subset of features, s.t.

1. The explanation is “probabilistically sufficient”
   
   Under the feature distribution, given the explanation, the classifier is likely to make the observed prediction.

2. It is minimal and “simple”

[Khosravi et al. IJCAI19, Wang et al. XXAI20]
Model-Based Algorithmic Fairness: FairPC

Learn classifier given
- features S and X
- training labels/decisions D

Group fairness by demographic parity:

*Fair decision $D_f$ should be independent of the sensitive attribute S*

Discover the **latent fair decision** $D_f$ by learning a PC.

[Choi et al. AAAI21]
Prediction with Missing Features

See work on

• Expected predictions / conformant learning [Khosravi et al.]
• Generative forests [Correia et al.]
Tractable Computation of Expected Kernels

- How to compute the expected kernel given two distributions $p, q$?

$$\mathbb{E}_{x \sim p, x' \sim q}[k(x, x')]$$

- Circuit representation for kernel functions, e.g., $k(x, x') = \exp \left( - \sum_{i=1}^{4} |X_i - X'_i|^2 \right)$
Tractable Computation of Expected Kernels: Applications

- Reasoning about support vector regression (SVR) with missing features

\[
\mathbb{E}_{x_m \sim p(X_m | x_o)} \left[ \sum_{i=1}^{m} w_i k(x_i, x) + b \right]
\]

missing features

SVR model

- Collapsed Black-box Importance Sampling: minimize kernelized Stein discrepancy

importance weights

\[
w^* = \arg\min_w \left\{ w^T K_{p,s} w \right\}
\sum_{i=1}^{n} w_i = 1, w_i \geq 0
\]

expected kernel matrix

tractability is a spectrum
Outline

1. What are tractable probabilistic circuits?
2. Are these models any good?
3. How far can we push tractable inference?
4. What is their expressive power?
Probabilistic circuits seem awfully general.

Are all tractable probabilistic models probabilistic circuits?
Enter: Determinantal Point Processes (DPPs)

DPPs are models where probabilities are specified by (sub)determinants

\[ L = \begin{bmatrix}
    1 & 0.9 & 0.8 & 0 \\
    0.9 & 0.97 & 0.96 & 0 \\
    0.8 & 0.96 & 1 & 0 \\
    0 & 0 & 0 & 1
\end{bmatrix} \]

- Tractable likelihoods and marginals
- Global Negative Dependence
- Diversity in recommendation systems

\[
\Pr_L(X_1 = 1, X_2 = 0, X_3 = 1, X_4 = 0) = \frac{1}{\det(L + I)} \det(L_{\{1,2\}})
\]
Relationship between PCs and DPPs

- Probabilistic Circuits
  - Positive Dependence
  - Fully Factorized
- Determinantal Point Processes
  - ?

We cannot tractably represent DPPs with subclasses of PCs

More Tractable

Deterministic and Decomposable PCs

Deterministic PCs with no negative parameters

Deterministic PCs with negative parameters

Decomposable PCs with no negative parameters (SPNs)

Decomposable PCs with negative parameters

Fewer Constraints

We don’t know

Probabilistic Generating Circuits

A Tractable Unifying Framework for PCs and DPPs

## Probability Generating Functions

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$\text{Pr}_\beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.02</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.08</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.12</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.48</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.02</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.08</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.04</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.16</td>
</tr>
</tbody>
</table>

$$g_\beta = \frac{0.16 z_1 z_2 z_3 + 0.04 z_1 z_2 + 0.08 z_1 z_3 + 0.02 z_1}{+ 0.48 z_2 z_3 + 0.12 z_2 + 0.08 z_3 + 0.02}.$$  

$$g_\beta = (0.1(z_1 + 1)(6z_2 + 1) - 0.4z_1 z_2)(0.8z_3 + 0.2)$$
Probabilistic Generating Circuits (PGCs)

\[ g_\beta = (0.1(z_1 + 1)(6z_2 + 1) - 0.4z_1z_2)(0.8z_3 + 0.2) \]

1. Sum nodes with weighted edges to children.
2. Product nodes with unweighted edges to children.
3. Leaf nodes: \( z_i \) or constant.
DPPs as PGCs

The generating polynomial for a DPP with kernel $L$ is given by:

$$g_L = \frac{1}{\det(L + I)} \det(I + L \text{diag}(z_1, \ldots, z_n)).$$

- Constant
- Division-free determinant algorithm (Samuelson-Berkowitz algorithm)

$g_L$ can be represented as a PGC of size $O(n^4)$
PGCs Support Tractable Likelihoods/Marginals

Purely symbolic

\[ z_i = \begin{cases} t, & X_i = 1 \\ 0, & X_i = 0 \\ 1, & \text{otherwise} \end{cases} \]

\[ \text{Pr}(X_1 = 1, X_2 = 0, \ldots) =? \]

\[ p(t) = \alpha_k t^k + \cdots + \alpha_1 t \]

\[ \alpha_k \text{ gives the answer} \]
Example

\[
\Pr(X_2 = 1, X_3 = 0) =? \\
\]

\[
\begin{array}{c|c|c|c}
X_1 & X_2 & X_3 & \Pr_\beta \\
\hline
0 & 0 & 0 & 0.02 \\
0 & 0 & 1 & 0.08 \\
0 & 1 & 0 & 0.12 \\
0 & 1 & 1 & 0.48 \\
1 & 0 & 0 & 0.02 \\
1 & 0 & 1 & 0.08 \\
1 & 1 & 0 & 0.04 \\
1 & 1 & 1 & 0.16 \\
\end{array}
\]
## Experiment Results: Amazon Baby Registries

SimplePGC achieves SOTA result on 11/15 datasets

<table>
<thead>
<tr>
<th></th>
<th>DPP</th>
<th>Strudel</th>
<th>EiNet</th>
<th>MT</th>
<th>SimplePGC</th>
</tr>
</thead>
<tbody>
<tr>
<td>bath</td>
<td>-8.55</td>
<td>-8.38</td>
<td>-8.49</td>
<td>-8.53</td>
<td><strong>-8.29</strong></td>
</tr>
<tr>
<td>bedding</td>
<td>-8.65</td>
<td>-8.50</td>
<td>-8.55</td>
<td>-8.59</td>
<td><strong>-8.41</strong></td>
</tr>
<tr>
<td>carseats</td>
<td>-4.74</td>
<td>-4.79</td>
<td>-4.72</td>
<td>-4.76</td>
<td><strong>-4.64</strong></td>
</tr>
<tr>
<td>diaper</td>
<td>-10.61</td>
<td>-9.90</td>
<td>-9.86</td>
<td>-9.93</td>
<td><strong>-9.72</strong></td>
</tr>
<tr>
<td>furniture</td>
<td>-4.38</td>
<td>-4.39</td>
<td>-4.38</td>
<td>-4.43</td>
<td><strong>-4.34</strong></td>
</tr>
<tr>
<td>gifts</td>
<td>-3.51</td>
<td><strong>3.39</strong></td>
<td>-3.42</td>
<td>-3.48</td>
<td>-3.47</td>
</tr>
<tr>
<td>health</td>
<td>-7.40</td>
<td>-7.37</td>
<td>-7.47</td>
<td>-7.49</td>
<td><strong>7.24</strong></td>
</tr>
<tr>
<td>media</td>
<td>-8.36</td>
<td><strong>7.62</strong></td>
<td>-7.82</td>
<td>-7.93</td>
<td>-7.69</td>
</tr>
<tr>
<td>moms</td>
<td>-3.55</td>
<td>-3.52</td>
<td><strong>3.48</strong></td>
<td>-3.54</td>
<td>-3.53</td>
</tr>
<tr>
<td>safety</td>
<td>-4.28</td>
<td>-4.43</td>
<td>-4.39</td>
<td>-4.36</td>
<td><strong>4.28</strong></td>
</tr>
<tr>
<td>strollers</td>
<td>-5.30</td>
<td>-5.07</td>
<td>-5.07</td>
<td>-5.14</td>
<td><strong>5.00</strong></td>
</tr>
<tr>
<td>toys</td>
<td>-8.05</td>
<td><strong>7.61</strong></td>
<td>-7.84</td>
<td>-7.88</td>
<td>-7.62</td>
</tr>
</tbody>
</table>
Conclusion

1. What are tractable probabilistic circuits?
2. Are these models any good?
3. How far can we push tractable inference?
4. What is their expressive power?
Learn more about probabilistic circuits?

Tutorial (3h)

[Image of tutorial slide]

Overview Paper (80p)

Probabilistic Circuits:
A Unifying Framework for Tractable Probabilistic Models

YooJung Choi
Antonio Vergari
Guy Van den Broeck
University of California, Los Angeles

Contents
1 Introduction 3
2 Probabilistic Inference: Models, Queries, and Tractability 4
  2.1 Probabilistic Models .................................. 5
  2.2 Probabilistic Queries .................................. 6
  2.3 Tractable Probabilistic Inference ..................... 8
  2.4 Properties of Tractable Probabilistic Models ........ 9

https://youtu.be/2RAG5-L9R70

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...for all leaves...

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

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

...and recurse over parents...
From BN trees to circuits
via compilation