Probabilistic Generating Circuits

Honghua Zhang, Brendan Juba and Guy Van den Broeck
Computer Science Department, University of California, Los Angeles
Computer Science Department, Washington University in St. Louis
{hzhang19, guyvdb} @cs.ucla.edu
bjuba@wustl.edu
Modeling Probability Distributions

Compute likelihoods

\[ \Pr(X_1 = 1, X_2 = 0, X_3 = 1) = ? \]

Table size exponential in # of variables!

<table>
<thead>
<tr>
<th>X_1</th>
<th>X_2</th>
<th>X_3</th>
<th>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>0.46</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.06</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>

Compute marginal probabilities

\[ \Pr(X_1 = 1, X_2 = 0) = ? \]
Tractable Probabilistic Models (TPMs)

**Expressive Efficiency:**
Size of the model efficient w/ respect to # of random variables

**Tractability:**
Probabilistic inference efficient w/ respect to size of the model
Contributions

1. Probability generating functions as probabilistic models.


3. PGCs support tractable computation of likelihoods and marginals.

4. PGCs are strictly more expressively efficient than two major families of TPMs: Probabilistic Circuits like SPNs and Determinantal Point Processes.

5. Show that PGCs have great potential in modeling real-world data.
It is natural to model distributions as *probability mass functions*;

but there are also *probability generating functions*...
### Probability Generating Functions


g_\beta = 0.16z_1z_2z_3 + 0.04z_1z_2 + 0.08z_1z_3 + 0.02z_1 + 0.48z_2z_3 + 0.12z_2 + 0.08z_3 + 0.02.

g_\beta = (0.1(z_1 + 1)(6z_2 + 1) - 0.4z_1z_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 $\oplus$ with weighted edges to children.
2. Product nodes $\otimes$ with unweighted edges to children.
3. Leaf nodes: $z_i$ or constant.
Likelihoods/Marginals for PGCs: Algorithm

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

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

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

Purely symbolic

\[ \alpha_k \text{ gives the answer} \]
Likelihoods/Marginals for PGCs: Example

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

\[
Pr(X_2 = 1, X_3 = 0) =
\begin{align*}
0.16t + 0.02 \\
0.2 \\
0.8 \\
0 \\
2 \\
1.0 \\
1 \\
1 \\
1 \\
0 \\
0 \\
0 \\
2 \\
0 \\
0.8t + 0.2 \\
0.2 \\
0.1 \\
-0.4 \\
12t + 2 \\
t \\
2 \\
6t + 1 \\
1 \\
1.0 \\
1.0 \\
6.0 \\
t \\
z_3 \\
z_2 \\
z_1 \\
z_1 \\
z_2 \\
X_1 & | X_2 & X_3 & Pr_\beta \\
0 & 0 & 0 & 0.02 \\
0 & 0 & 1 & 0.08 \\
0 & 1 & 0 & 0.12 \\
0 & 1 & 1 & 0.48 \\
1 & 0 & 0 & 0.02 \\
1 & 0 & 1 & 0.08 \\
1 & 1 & 0 & 0.04 \\
1 & 1 & 1 & 0.16
\end{align*}
\]
Likelihoods/Marginals for PGCs: Tractability

PGC of size $r$
$n$ binary random variables

Feed-forward Pass $+$ At each node: Sum/Product of degree-$n$ polynomials

Time Complexity: $O(rP(n))$

$P(n):$ # of operations to multiply two degree-$n$ polynomials
Okay... PGCs are **tractable** models

Are PGCs **expressively efficient**?
Other TPMs Can be Tractably Represented as PGCs
Probabilistic Circuits (PCs)

- $X_i$ and $\overline{X_i}$ instead of $z_i$
- Probability Mass Functions
- Only tractable when decomposable
- Sum-Product Networks (SPNs)
- Graphical Models w/ bounded treewidth
Decomposable PCs as PGCs

- Polynomial-time algorithm for “smoothing” decomposable PCs
- Set $X_i$ to $z_i$
- Set $\overline{X_i}$ to 1
- Decomposable PC
- Smooth & Decomposable PC (SPN)
- PGC
Determinantal Point Processes (DPPs)

\[
L^\beta = \begin{bmatrix}
1 & 2 & 0 \\
2 & 6 & 0 \\
0 & 0 & 4 \\
\end{bmatrix}
\]

\[
\Pr_L(X_1 = 1, X_2 = 0, X_3 = 1) \propto \begin{vmatrix}
1 & 0 \\
0 & 4 \\
\end{vmatrix} = 4 \quad \text{Normalizing Constant: } \det(L^\beta + I) = 50
\]

Tractable likelihoods and marginals

Global Negative Dependence

Widely used in recommendation systems
The generating polynomial for a DPP with kernel $L$ is given by:

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

$g_L$ can be represented as a PGC of size $O(n^4)$.
PGCs are **Strictly** More Expressively Efficient

(a) PGCs are strictly more expressive than DPPs

(b) PGCs are strictly more expressively efficient than decomposable PCs including SPNs
Beyond Decomposable PCs and DPPs

PGCs

Decomposable PCs

DPPs

Determinantal PGCs
Compositional Operations for PGCs

(a) Weighted Sum

(b) Product

(c) Hierarchical Composition
Determinantal PGC (DetPGC): a Non-trivial Toy Model

$g_{\delta} = \det(I + L \text{ diag}(f_1, f_2, \ldots, f_m))$

- Take the hierarchical composition of a DPP and $Pr_i$'s
- Model each subgroup of variables with a PGC $Pr_i$ with generating function $f_i$
- Partition the random variables into subgroups
We know that PGCs are more expressively efficient in theory...

As a proof-of-concept, let’s try our toy model on real-world datasets
Experiments: Density Estimation

Goal:
Higher average log-likelihood

Benchmark #1:
Twenty Datasets

Benchmark #2:
Amazon Baby Registries
Experiments: Model & Baselines

SimplePGCs: mixtures over DetPGCs

Fixed-structure Models: (w/o structure learning)
- DPPs
- Einsum Networks

Models w/ Structure Learning:
- Strudel
- Mixtures of Trees
Experiment Results: Twenty Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DPP</th>
<th>Strudel</th>
<th>EiNet</th>
<th>MT</th>
<th>SimplePGC</th>
</tr>
</thead>
<tbody>
<tr>
<td>nltcs</td>
<td>-9.23</td>
<td>-6.07</td>
<td>-6.02</td>
<td>-6.01</td>
<td>-6.05*</td>
</tr>
<tr>
<td>msnbc</td>
<td>-6.48</td>
<td>-6.04</td>
<td>-6.12</td>
<td>-6.07</td>
<td>-6.06†</td>
</tr>
<tr>
<td>kdd</td>
<td>-2.45</td>
<td>-2.14</td>
<td>-2.18</td>
<td>-2.13</td>
<td>-2.14†</td>
</tr>
<tr>
<td>audio</td>
<td>-49.31</td>
<td>-42.20</td>
<td>-39.88</td>
<td>-40.08</td>
<td>-40.21*</td>
</tr>
<tr>
<td>jester</td>
<td>-63.88</td>
<td>-54.24</td>
<td>-52.56</td>
<td>-53.08</td>
<td>-53.54*</td>
</tr>
<tr>
<td>netflix</td>
<td>-64.18</td>
<td>-57.93</td>
<td>-56.54</td>
<td>-56.74</td>
<td>-57.42*</td>
</tr>
<tr>
<td>accidents</td>
<td>-35.61</td>
<td>-29.05</td>
<td>-35.59</td>
<td>-29.63</td>
<td>-30.46†</td>
</tr>
<tr>
<td>retail</td>
<td>-11.43</td>
<td>-10.83</td>
<td>-10.92</td>
<td>-10.83</td>
<td>-10.84†</td>
</tr>
<tr>
<td>pumsb</td>
<td>-51.98</td>
<td>-24.39</td>
<td>-31.95</td>
<td>-23.71</td>
<td>-29.56†</td>
</tr>
<tr>
<td>dna</td>
<td>-82.19</td>
<td>-87.15</td>
<td>-96.09</td>
<td>-85.14</td>
<td>-80.82†</td>
</tr>
<tr>
<td>kosarek</td>
<td>-13.35</td>
<td>-10.70</td>
<td>-11.03</td>
<td>-10.62</td>
<td>-10.72†</td>
</tr>
<tr>
<td>msweb</td>
<td>-11.31</td>
<td><strong>9.74</strong></td>
<td>-10.03</td>
<td>-9.85</td>
<td>-9.98†</td>
</tr>
<tr>
<td>book</td>
<td>-41.22</td>
<td>-34.49</td>
<td>-34.74</td>
<td>-34.63</td>
<td>-34.11†</td>
</tr>
<tr>
<td>movie</td>
<td>-83.55</td>
<td>-53.72</td>
<td><strong>51.71</strong></td>
<td>-54.60</td>
<td>-53.15*</td>
</tr>
<tr>
<td>webkb</td>
<td>-180.61</td>
<td><strong>154.83</strong></td>
<td>-157.28</td>
<td>-156.86</td>
<td>-155.23†</td>
</tr>
<tr>
<td>reuters</td>
<td>-107.44</td>
<td>-86.35</td>
<td>-87.37</td>
<td><strong>85.90</strong></td>
<td>-87.65</td>
</tr>
<tr>
<td>20ng</td>
<td>-174.43</td>
<td><strong>153.87</strong></td>
<td>-153.94</td>
<td>-154.24</td>
<td>-154.03°</td>
</tr>
<tr>
<td>bbc</td>
<td>-278.15</td>
<td>-256.53</td>
<td><strong>248.33</strong></td>
<td>-261.84</td>
<td>-254.81°</td>
</tr>
<tr>
<td>ad</td>
<td>-63.20</td>
<td>-16.52</td>
<td>-26.27</td>
<td>-16.02</td>
<td>-21.65†</td>
</tr>
</tbody>
</table>

SimplePGC beats the fixed-structure baselines on 12/20 datasets

SimplePGC beats the baselines with structure learning on 4/20 datasets

SimplePGC achieves SOTA result on 2/20 datasets
## Experiment Results: Amazon Baby Registries

<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>-8.29*</td>
</tr>
<tr>
<td>bedding</td>
<td>-8.65</td>
<td>-8.50</td>
<td>-8.55</td>
<td>-8.59</td>
<td>-8.41*</td>
</tr>
<tr>
<td>carseats</td>
<td>-4.74</td>
<td>-4.79</td>
<td>-4.72</td>
<td>-4.76</td>
<td>-4.64*</td>
</tr>
<tr>
<td>feeding</td>
<td>-11.86</td>
<td>-11.42</td>
<td>-11.27</td>
<td>-11.30</td>
<td>-11.17*</td>
</tr>
<tr>
<td>furniture</td>
<td>-4.38</td>
<td>-4.39</td>
<td>-4.38</td>
<td>-4.43</td>
<td>-4.34*</td>
</tr>
<tr>
<td>gifts</td>
<td>-3.51</td>
<td>3.39</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>-7.24*</td>
</tr>
<tr>
<td>media</td>
<td>-8.36</td>
<td>-7.62</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>-3.48</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>-4.28*</td>
</tr>
<tr>
<td>strollers</td>
<td>-5.30</td>
<td>-5.07</td>
<td>-5.07</td>
<td>-5.14</td>
<td>-5.00*</td>
</tr>
<tr>
<td>toys</td>
<td>-8.05</td>
<td>-7.61</td>
<td>-7.84</td>
<td>-7.88</td>
<td>-7.62*</td>
</tr>
</tbody>
</table>

**SimplePGC** beats the fixed-structure baselines on 13/15 datasets.

**SimplePGC** beats the baselines with structure learning on 11/15 datasets.

**SimplePGC** achieves SOTA result on 11/15 datasets.
1. A new class of TPMs called Probabilistic Generating Circuits that support tractable computation of likelihoods and marginals.

2. PGCs are strictly more expressively efficient than decomposable Probabilistic Circuits including SPNs and Determinantal Point Processes.

3. Experiments show that PGCs have great potential in modeling real-world data.
1. Development of architecture search algorithm for PGCs.

2. Strongly Rayleigh distributions and PGCs:
   ---What subclasses of strongly Rayleigh distributions can be represented as polynomial-size PGCs?
Future Work #2: Searching for PGC Structures

1. Development of architecture search algorithm for PGCs.

2. Design PGC structures that work well on specialized tasks.
Probabilistic Generating Circuits

Honghua Zhang 1  Brendan Juba 2  Guy Van den Broeck 1

Abstract
Generating functions, which are widely used in combinatorics and probability theory, encode function values into the coefficients of a polynomial. In this paper, we explore their use as a tractable probabilistic model, and propose probabilistic generating circuits (PGCs) for their efficient representation. PGCs are strictly more expressive efficient than many existing tractable probabilistic models, including determinantal point processes (DPPs), probabilistic circuits (PCs) such as sum-product networks, and tractable graphical models. We contend that PGCs are not just a theoretical framework that unifies vastly different existing models, but also show great potential in modeling realistic data. We exhibit a simple class of PGCs that are not trivially subsumed by simple combinations of PCs and DPPs, and obtain competitive performance on a suite of density estimation benchmarks. We also highlight PGCs’ connection to the theory of strongly Rayleigh distributions.

1. Introduction
Probabilistic modeling is an important task in machine learning. Scaling up such models is a key challenge: probabilistic inference quickly becomes intractable as the models become large and sophisticated (Roth, 1996). Central to this effort is the development of tractable probabilistic models (TPMs) that guarantee tractable probabilistic inference in the size of the model, yet can efficiently represent a wide range of probability distributions. There have been a modification (Wieche, 2009; Kisa et al., 2014; Vergari et al., 2020) such as sum-product networks (Poon & Domingos, 2011).

Ideally, we want our probabilistic models to be as expressive efficient (Martens & Medalalimi, 2014) as possible, meaning that they can efficiently represent as many classes of distributions as possible, and adapt to a wider spectrum of realistic applications. Often, however, stronger expressive power comes at the expense of tractability: fewer restrictions can make a model more expressive efficient, but it can also make probabilistic inference intractable. We therefore raise the following central research question of this paper: Does there exist a class of tractable probabilistic models that is strictly more expressive efficient than current TPMs?

All aforementioned models are usually seen as representing probability mass functions: they take assignments to random variables as input and output likelihoods. In contrast, especially in the field of probability theory, it is also common to represent distributions as probability generating polynomials (or generating polynomials for short). Generating polynomials are a powerful mathematical tool, but they have not yet found direct use as a probabilistic machine learning representation that permits tractable probabilistic inference.

We make the key observation that the marginal probabilities (including likelihoods) for a probability distribution can be computed by evaluating its generating polynomial in a particular way. Based on this observation, we propose probabilistic generating circuits (PGCs), a class of probabilistic models that represent probability generating polynomials compactly as directed acyclic graphs. PGCs provide a partly positive answer to our research question: they are the first known class of TPMs that are strictly more expressive efficient than decomposable probabilistic circuits (PCs), in particular, sum-product networks and determinantal point

Thank you!

Honghua Zhang
hzhang19@cs.ucla.edu