Probabilistic Generating Circuits

Honghua Zhang[1] Brendan Juba[2] Guy Van den Broeck[1]
[1]University of California, Los Angeles [2]Washington University in St. Louis

Probability Distribution

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

Probability Generating Function*

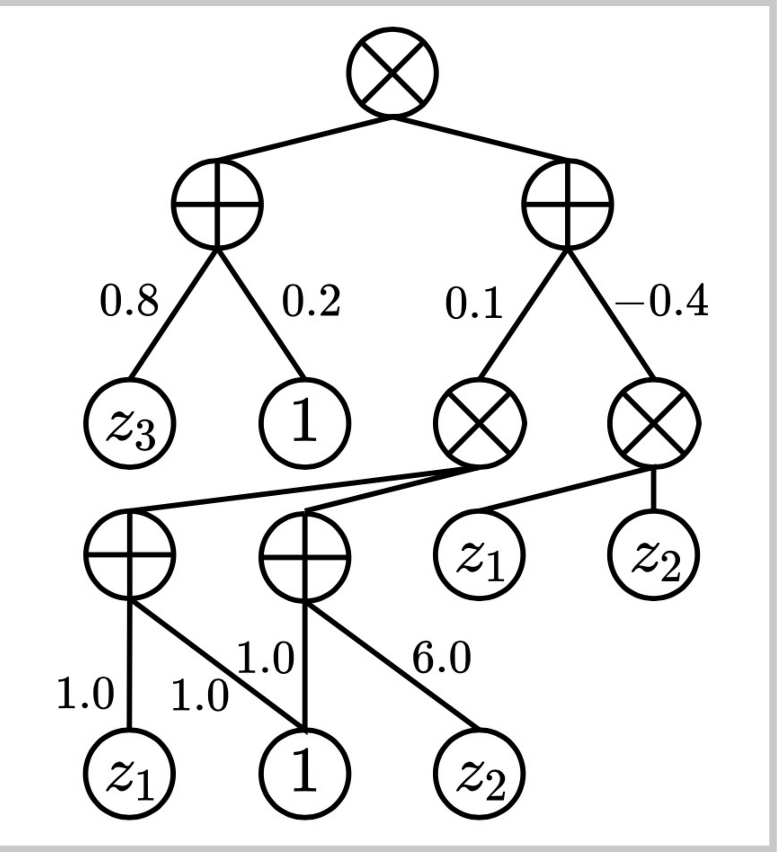
$$g_\beta = 0.16z_1z_2z_3 + 0.04z_1z_2 + 0.08z_1z_3 + 0.02z_1 + 0.48z_2z_3 + 0.12z_2 + 0.08z_3 + 0.02.$$

Probability Generating Function (Compact Form)

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

*In Mathematics, distributions over discrete random variables are often represented as probability generating functions (PGFs): each term corresponds to one possible assignment and the coefficients give the corresponding probabilities.

PGC



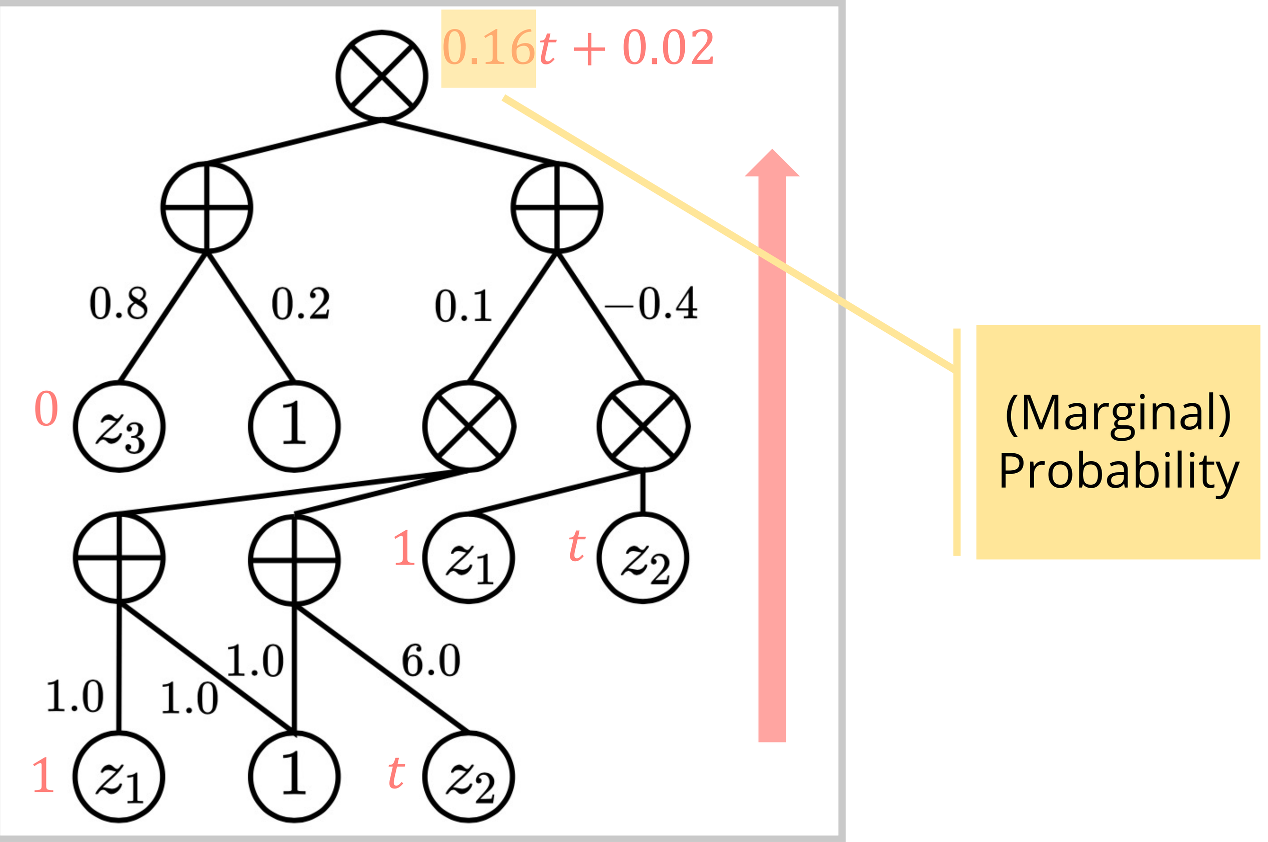
Probabilistic Inference

$\Pr(X_2 = 1, X_3 = 0) = ?$

Plugging in

$$z_i = \begin{cases} t, & X_i = 1 \\ 0, & X_i = 0 \\ 1, & otherwise \end{cases}$$

Feedforward Pass



What are TPMs?

Tractable Probabilistic Models (TPMs) are models for probability distributions such that:

- The size of the model is efficient with respect to the # of random variables. (expressively efficient)
- Probabilistic inference is efficient with respect to the size of the model. (tractable)

Contributions

- We study the use of probability generating functions (PGFs) as probabilistic models.
- We propose a new class of TPMs called Probabilistic Generating Circuits (PGCs) to represent PGFs efficiently.
- PGCs support *tractable marginals/likelihoods* and are *strictly* more expressively efficient than other TPMs including decomposable probabilistic circuits (PCs) [1] like *sum-product networks* (SPNs) [2] and *determinantal point processes* (DPPs) [3].

PGC

PGCs represent PGFs as directed acyclic graphs (DAGs), and they contain three types of nodes:

- Sum nodes ⊕ with weighted edges to children.
- Product nodes ⊗ with un-weighted edges to children.
- Leaf nodes: z_i or constant.

Other TPMs as PGCs

Decomposable PCs (SPNs) as PGCs

Given a smooth and decomposable PC (SPN), by replacing its leaf variables accordingly, we immediately obtain a PGC that represents the same distribution.

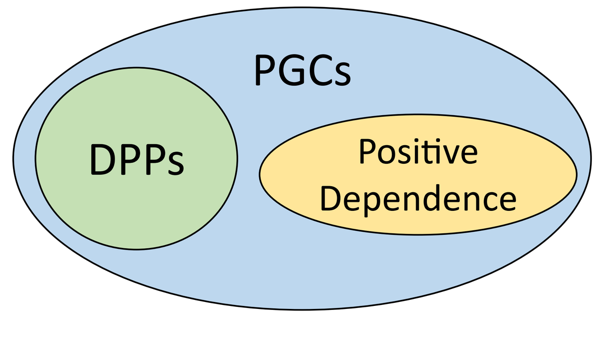
DPPs as PGCs

- Given a DPP (L-ensemble) with kernel matrix L, its probability generating function is given by [4]:

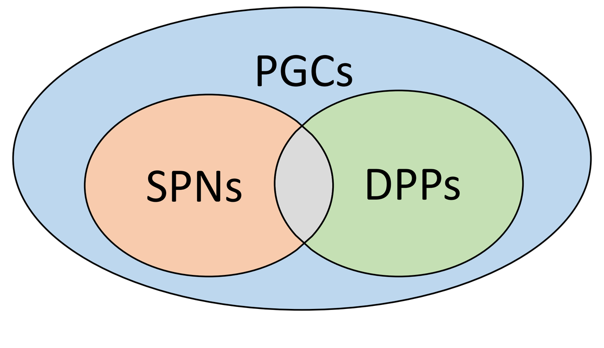
$$g_L = \frac{1}{\det(L+I)} \det(I + L\text{diag}(z_1, \dots, z_n)).$$

- By encoding a polynomial-time division-free determinant algorithm [5] as PGC, we obtain a polynomial-size PGC that represents the DPP.

PGCs are strictly more expressively efficient

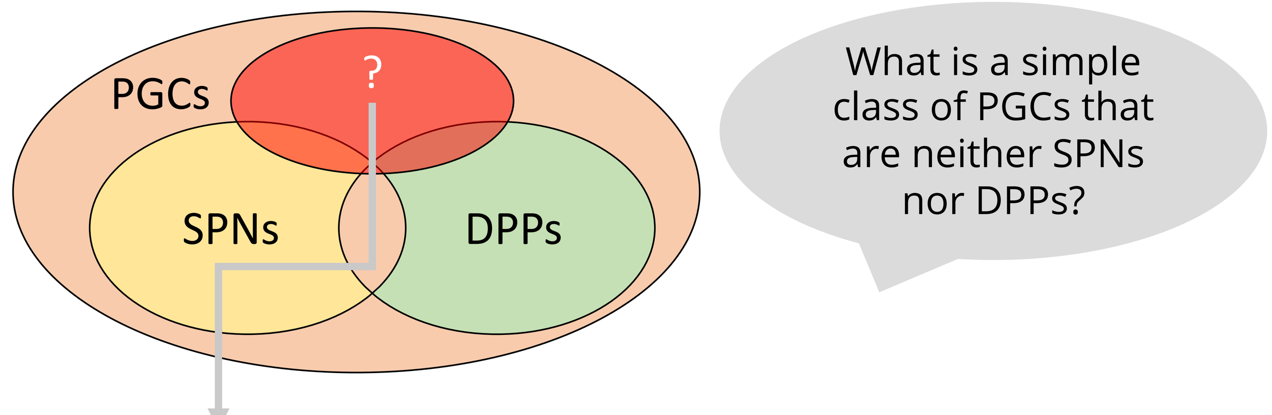


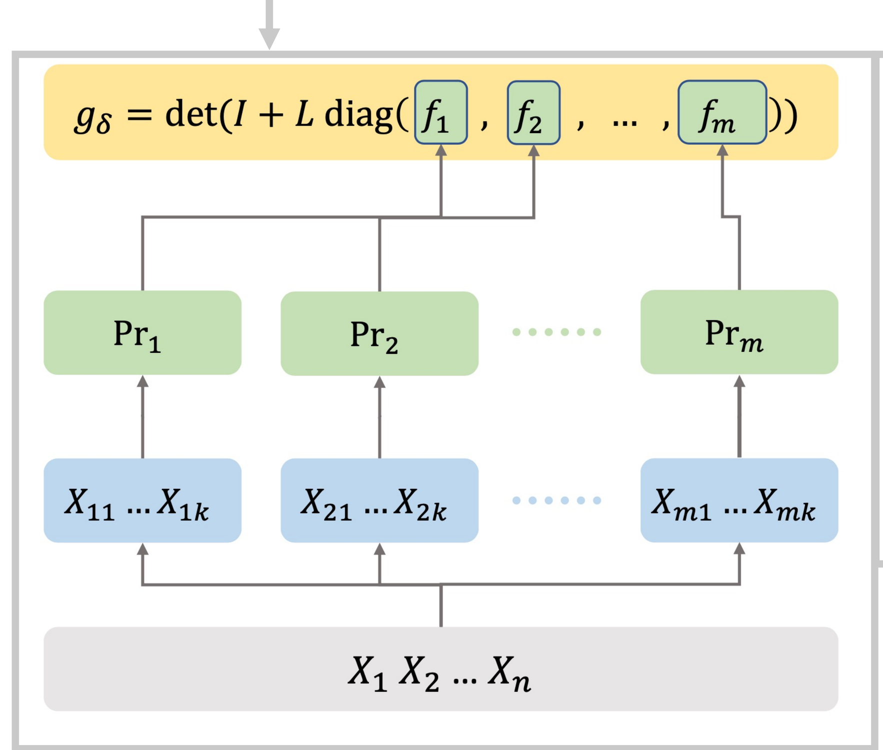
(a) PGCs are *strictly* more expressive than DPPs



(a) PGCs are *strictly* more expressively efficient than SPNs.

A Non-trivial Example





Determinantal PGC:

- DetPGC is constructed by taking the hierarchical composition* of a DPP (represented as a PGC) and several "smaller" children PGCs.
- DetPGC is not trivially subsumed by simple combinations of SPNs and DPPs (e.g., SPNs with DPPs as leaves)

*Hierarchical Composition:

Given a "parent" PGC $g(z_1, \dots, z_n)$ and "child" PGCs $f_1, \dots, f_n$ on distinct variables, the composition $g(f_1, \dots, f_n)$ is a valid PGC.

Experiments: a proof-of-concept

- Density estimation benchmarks: the Twenty Datasets, the Amazon Baby Registries.
- Model: SimplePGCs, which are mixtures over DetPGCs.
- Baselines: DPPs, Strudel, Einsum Networks, Mixture of Trees.

	DPP	Strudel	EiNet	MT	SimplePGC
nltcs	−9.23	−6.07	−6.02	**−6.01**	-6.05^{*}
msnbc	−6.48	**−6.04**	−6.12	−6.07	$-6.06^{\dagger\circ}$
kdd	−2.45	−2.14	−2.18	**−2.13**	$-2.14^{*\dagger}$
plants	−31.20	−13.22	−13.68	**−12.95**	$-13.52^{\dagger}$
audio	−49.31	−42.20	**−39.88**	−40.08	-40.21^{*}
jester	−63.88	−54.24	**−52.56**	−53.08	-53.54^{*}
netflix	−64.18	−57.93	**−56.54**	−56.74	-57.42^{*}
accidents	−35.61	**−29.05**	−35.59	−29.63	$-30.46^{\dagger}$
retail	−11.43	**−10.83**	−10.92	**−10.83**	$-10.84^{\dagger}$
pumsb	−51.98	−24.39	−31.95	**−23.71**	$-29.56^{\dagger}$
dna	−82.19	−87.15	−96.09	−85.14	$\mathbf{-80.82}^{*\dagger\circ}$
kosarek	−13.35	−10.70	−11.03	**−10.62**	$-10.72^{\dagger}$
msweb	−11.31	**−9.74**	−10.03	−9.85	$-9.98^{\dagger}$
book	−41.22	−34.49	−34.74	−34.63	$\mathbf{-34.11}^{*\dagger\circ}$
movie	−83.55	−53.72	**−51.71**	−54.60	$-53.15^{*\circ}$
webkb	−180.61	**−154.83**	−157.28	−156.86	$-155.23^{\dagger\circ}$
reuters	−107.44	−86.35	−87.37	**−85.90**	−87.65
20ng	−174.43	**−153.87**	−153.94	−154.24	-154.03°
bbc	−278.15	−256.53	**−248.33**	−261.84	$-254.81^{*\circ}$
ad	−63.20	−16.52	−26.27	**−16.02**	$-21.65^{\dagger}$

(a) Average log-likelihoods on the Twenty Datasets

	DPP	Strudel	EiNet	MT	SimplePGC
apparel	−9.88	−9.51	−9.24	−9.31	$\mathbf{-9.10}^{*\dagger\circ}$
bath	−8.55	−8.38	−8.49	−8.53	$\mathbf{-8.29}^{*\dagger\circ}$
bedding	−8.65	−8.50	−8.55	−8.59	$\mathbf{-8.41}^{*\dagger\circ}$
carseats	−4.74	−4.79	−4.72	−4.76	$\mathbf{-4.64}^{*\dagger\circ}$
diaper	−10.61	−9.90	−9.86	−9.93	$\mathbf{-9.72}^{*\dagger\circ}$
feeding	−11.86	−11.42	−11.27	−11.30	$\mathbf{-11.17}^{*\dagger\circ}$
furniture	−4.38	−4.39	−4.38	−4.43	$\mathbf{-4.34}^{*\dagger\circ}$
gear	−9.14	−9.15	−9.18	−9.23	$\mathbf{-9.04}^{*\dagger\circ}$
gifts	−3.51	**−3.39**	−3.42	−3.48	-3.47°
health	−7.40	−7.37	−7.47	−7.49	$\mathbf{-7.24}^{*\dagger\circ}$
media	−8.36	**−7.62**	−7.82	−7.93	$-7.69^{\dagger\circ}$
moms	−3.55	−3.52	**−3.48**	−3.54	-3.53°
safety	−4.28	−4.43	−4.39	−4.36	$\mathbf{-4.28}^{*\dagger\circ}$
strollers	−5.30	−5.07	−5.07	−5.14	$\mathbf{-5.00}^{*\dagger\circ}$
toys	−8.05	**−7.61**	−7.84	−7.88	$-7.62^{\dagger\circ}$

(b) Average log-likelihoods on the Amazon Baby Registries.

Table 1. Bold numbers indicate the best log-likelihood. For SimplePGC, annotations $*$, † and ∘ mean better log-likelihood compared to Strudel, EiNet and MT, respectively.

Code available at → **Paper available at →**

ICML 2021

References: [1] Vergari, A., Choi, Y., Peharz, R., and Van den Broeck, G. Probabilistic circuits: Representations, inference, learning and applications. Tutorial at the The 34th AAAI Conference on Artificial Intelligence, 2020.
[2] Poon, H. and Domingos, P. Sum-product networks: A new deep architecture. In *ICCV Workshops*, pp. 689–690, 2011.
[3] Kulesza, A. and Taskar, B. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2–3):123–286, 2012.
[4] Borcea, J., Branden, P., and Liggett, T. Negative dependence and the geometry of polynomials. *AMS*, 22(2):521–567, 2009.
[5] Samuelson, P. A. A method of determining explicitly the coefficients of the characteristic equation. *The Annals of Mathematical Statistics*, 13(4):424–429, 1942.

Acknowledgements: This work is partially supported by NSF grants #IIS-1943641, #IIS-1633857, #CCF-1837129, #CCF-1718380, #IIS-1908287, and #IIS-1939677, DARPA XAI grant #N66001-17-2-4032, Sloan and UCLA Samueli Fellowships, and gifts from Intel and Facebook Research.