From Probabilistic Circuits to Probabilistic Programs and Back

Guy Van den Broeck

PROBPROG - Oct 24, 2020
Probabilistic graphical models is how we do probabilistic AI!

Graphical models of variable-level (in)dependence are a broken abstraction.

[VdB KRR15]
Trying to be provocative

Probabilistic graphical models is how we do probabilistic AI!

Graphical models of variable-level (in)dependence are a broken abstraction.

3.14 Smokes(x) \land Friends(x,y) \Rightarrow Smokes(y)
Trying to be provocative

 Probabilistic graphical models is how we do probabilistic AI!

 Graphical models of variable-level (in)dependence are a broken abstraction.

 Bean Machine

\[ \mu_k \sim \text{Normal}(\alpha, \beta) \]
\[ \sigma_k \sim \text{Gamma}(\nu, \rho) \]
\[ \theta_k \sim \text{Dirichlet}(\kappa) \]
\[ x_i \sim \begin{cases} \text{Categorical}(\text{init}) & \text{if } i = 0 \\ \text{Categorical}(\theta_{x_{i-1}}) & \text{if } i > 0 \end{cases} \]
\[ y_i \sim \text{Normal}(\mu_{x_i}, \sigma_{x_i}) \]

[Tehrani et al. PGM20]
Computational Abstractions

Let us think of probability distributions as objects that are computed.

Abstraction = Structure of Computation

Two examples:

2. Probabilistic Programs

*eyeroll*
Computational Abstractions

Let us think of probability distributions as objects that are computed.

Abstraction = Structure of Computation

Two examples:
1. Probabilistic Circuits
2. Probabilistic Programs
Probabilistic Circuits
The Alphabet Soup of probabilistic models
Intractable and tractable models
"Every talk needs a joke and a literature overview slide, not necessarily distinct"
- after Ron Graham
a unifying framework for tractable models
Input nodes $c$ are tractable (simple) distributions, e.g., univariate gaussian or indicator $p_c(X=1) = [X=1]$
Product nodes are factorizations $\prod_{c \in \text{in}(n)} p_c(x)$
Sum nodes are mixture models $\sum_{c \in \text{in}(n)} \theta_{n,c} p_c(x)$
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

[Darwiche & Marquis JAIR 2001, Poon & Domingos UAI11]
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

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

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

- leafs over \( X_1 \) and \( X_3 \) output \( Z_i = \int p(x_i) \, dx_i \)
  \[ \Rightarrow \text{for normalized leaf distributions: 1.0} \]
- leafs over \( X_2 \) and \( X_4 \) output EVI
- feedforward evaluation (bottom-up)
<table>
<thead>
<tr>
<th>Property</th>
<th>MAR</th>
<th>CON</th>
<th>MOM</th>
<th>MAP</th>
<th>MMAP</th>
<th>ENT</th>
<th>DIV</th>
<th>EXP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smoothness</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Decomposability</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consistency</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Determinism</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Marginal determinism</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structured decomposability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paired str. decomposability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
tractability is a spectrum
<table>
<thead>
<tr>
<th>Model</th>
<th>smooth</th>
<th>dec.</th>
<th>det.</th>
<th>str.dec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic Circuits (ACs)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✗️</td>
</tr>
<tr>
<td>Sum-Product Networks (SPNs)</td>
<td>✔️</td>
<td>✔️</td>
<td>✗️</td>
<td>✗️</td>
</tr>
<tr>
<td>Cutset Networks (CNets)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✗️</td>
</tr>
<tr>
<td>Probabilistic Decision Graphs</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>(Affine) ADDs</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>AndOrGraphs</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>PSDDs</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Expressive models without compromises
### How expressive are probabilistic circuits?

**density estimation benchmarks**

<table>
<thead>
<tr>
<th>dataset</th>
<th>best circuit</th>
<th>BN</th>
<th>MADE</th>
<th>VAE</th>
<th>dataset</th>
<th>best circuit</th>
<th>BN</th>
<th>MADE</th>
<th>VAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>nltcs</td>
<td><strong>-5.99</strong></td>
<td>-6.02</td>
<td>-6.04</td>
<td><strong>-5.99</strong></td>
<td>dna</td>
<td><strong>-79.88</strong></td>
<td>-80.65</td>
<td>-82.77</td>
<td>-94.56</td>
</tr>
<tr>
<td>msnbc</td>
<td><strong>-6.04</strong></td>
<td><strong>-6.04</strong></td>
<td>-6.06</td>
<td>-6.09</td>
<td>kosarek</td>
<td><strong>-10.52</strong></td>
<td>-10.83</td>
<td>-</td>
<td><strong>-10.64</strong></td>
</tr>
<tr>
<td>plants</td>
<td><strong>-11.84</strong></td>
<td>12.65</td>
<td>-12.32</td>
<td>-12.34</td>
<td>book</td>
<td><strong>-33.82</strong></td>
<td>-36.41</td>
<td>-33.95</td>
<td><strong>-33.19</strong></td>
</tr>
<tr>
<td>audio</td>
<td>-39.39</td>
<td>-40.50</td>
<td>-38.95</td>
<td><strong>-38.67</strong></td>
<td>movie</td>
<td>-50.34</td>
<td>-54.37</td>
<td>-48.7</td>
<td><strong>-47.43</strong></td>
</tr>
<tr>
<td>jester</td>
<td>-51.29</td>
<td><strong>-51.07</strong></td>
<td>-52.23</td>
<td>-51.54</td>
<td>webkb</td>
<td>-149.20</td>
<td>-157.43</td>
<td>-149.59</td>
<td><strong>-146.9</strong></td>
</tr>
<tr>
<td>netflix</td>
<td>-55.71</td>
<td>-57.02</td>
<td>-55.16</td>
<td><strong>-54.73</strong></td>
<td>cr52</td>
<td>-81.87</td>
<td>-87.56</td>
<td>-82.80</td>
<td><strong>-81.33</strong></td>
</tr>
<tr>
<td>accidents</td>
<td>-26.89</td>
<td><strong>-26.32</strong></td>
<td>-26.42</td>
<td>-29.11</td>
<td>c20ng</td>
<td>-151.02</td>
<td>-158.95</td>
<td>-153.18</td>
<td><strong>-146.9</strong></td>
</tr>
<tr>
<td>retail</td>
<td><strong>-10.72</strong></td>
<td>-10.87</td>
<td>-10.81</td>
<td>-10.83</td>
<td>bbc</td>
<td><strong>-229.21</strong></td>
<td>-257.86</td>
<td>-242.40</td>
<td>-240.94</td>
</tr>
<tr>
<td>pumbs*</td>
<td>-22.15</td>
<td><strong>-21.72</strong></td>
<td>-22.3</td>
<td>-25.16</td>
<td>ad</td>
<td>-14.00</td>
<td>-18.35</td>
<td><strong>-13.65</strong></td>
<td>-18.81</td>
</tr>
</tbody>
</table>
The diagram illustrates the trade-off between tractability and expressiveness of various models. Models such as Fully factorized trees, NB Polytrees, LTM Mixtures, and TJT are located in the less tractable queries and less expressive efficient quadrant. Models like PCs, PCs, PCs, PCs, NADEs, BNs, NFs, VAEs, MNs, and GANs are in the more tractable queries and more expressive efficient quadrant.
Want to learn more?

Tutorial (3h)

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

Overview Paper (80p)

Probabilistic Circuits:
A Unifying Framework for Tractable Probabilistic Models*

YooJung Choi
Antonio Vergari
Guy Van den Broeck

Computer Science Department
University of California
Los Angeles, CA, USA

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

Training PCs in Julia with Juice.jl

Training maximum likelihood parameters of probabilistic circuits

```julia
julia> using ProbabilisticCircuits;
julia> data, structure = load(...);
```

```
num_examples(data)
17,412
```

```
num_edges(structure)
270,448
```

```
@btime estimate_parameters(structure, data);
63 ms
```

Custom SIMD and CUDA kernels to parallelize over layers and training examples.
Probabilistic circuits seem awfully general.

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

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

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

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

Computing marginal probabilities is tractable.

[Zhang et al. UAI20]
We cannot tractably represent DPPs with classes of PCs … yet

**An almost universal tractable language**

**Stay Tuned!**

[Zhang et al. UAI20; Martens & Medabalimi Arxiv15]
The AI Dilemma

Pure Logic → Pure Learning
The AI Dilemma

Pure Logic

• Slow thinking: deliberative, cognitive, model-based, extrapolation
• Amazing achievements until this day
• “Pure logic is brittle”
  noise, uncertainty, incomplete knowledge, …
The AI Dilemma

Pure Logic

- Fast thinking: instinctive, perceptive, model-free, interpolation
- Amazing achievements recently
- “Pure learning is brittle”

Pure Learning

- Bias, algorithmic fairness, interpretability, explainability, adversarial attacks, unknown unknowns, calibration, verification, missing features, missing labels, data efficiency, shift in distribution, general robustness and safety
- Fails to incorporate a sensible model of the world
“Pure learning is brittle”

bias, algorithmic fairness, interpretability, explainability, adversarial attacks, unknown unknowns, calibration, verification, missing features, missing labels, data efficiency, shift in distribution, general robustness and safety

text: fails to incorporate a sensible model of the world
Prediction with Missing Features

Train Classifier

Test with missing features
Expected Predictions

Consider all possible complete inputs and reason about the expected behavior of the classifier

\[
E_{x^m \sim p(x^m | x^o)} \left[ f(x^m x^o) \right]
\]

\[x^o = \text{observed features}\]
\[x^m = \text{missing features}\]

Experiment:
- \(f(x) = \text{logistic regres.}\)
- \(p(x) = \text{naive Bayes}\)

[Khosravi et al. IJCAI19, NeurIPS20, Artemiss20]
What about complex feature distributions?

- feature distribution is a probabilistic circuits
- classifier is a compatible regression circuit

Recursion that “breaks down” the computation.

Expectation of function $m$ w.r.t. dist. $n$?

Solve subproblems: $(1,3), (1,4), (2,3), (2,4)$

[Khosravi et al. IJCAI19, NeurIPS20, Artemiss20]
Probabilistic Circuits for Missing Data

[Khosravi et al. IJCAI19, NeurIPS20, Artemiss20]
using ProbabilisticCircuits
pc = load_prob_circuit(zoo_psdd_file("insurance.psdd"));
rc = load_logistic_circuit(zoo_lc_file("insurance.circuit"), 1);

Is the predictive model biased by gender?

groups = make_observations([["male"], ["female")])
exps, _ = Expectation(pc, rc, groups);
println("Female : \$ $(exps[2])");
println("Male : \$ $(exps[1])");
println("Diff : \$ $(exps[2] - exps[1])");
Female : $ 14170.125469335406
Male : $ 13196.548926381849
Diff : $ 973.5765429535568
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. Arxiv20]
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]
"Pure learning is brittle"

- bias, **algorithmic fairness**, interpretability, **explainability**, adversarial attacks,
- unknown unknowns, calibration, verification, **missing features**, missing labels,
- data efficiency, shift in distribution, general robustness and safety

We need to incorporate a sensible probabilistic model of the world
Probabilistic Programs
**Dice** probabilistic programming language

http://dicelang.cs.ucla.edu/

https://github.com/SHoltzen/dice

Talk in 25min

Holtzen et al. OOPSLA20

---

dice is a probabilistic programming language focused on fast exact inference for discrete probabilistic programs. For more information on dice, see the about page.

Below is an online dice code demo. To run the example code, press the "Run" button.

```plaintext
1 fun sendChar(key: Int, observation: Int) {
2   let gen = discrete(0.25, 0.25, 0.125, 0.125) in // sample a Foolang character
3   let enc = key + gen in // encrypt the character
4   observe observation == enc
5 }

// sample a uniform random key: A-0, B-1, C-2, D-3
8 let key = discrete(0.25, 0.25, 0.25, 0.25) in
9 // observe the ctienertt CCC
11 let tmp = sendChar(key, 2) in
12 let tmp = sendChar(key, 5) in
14 let tmp = sendChar(key, 3) in
15 let tmp = sendChar(key, 2) in
16 17 key
```
Symbolic Compilation to Probabilistic Circuits

- Probabilistic Program
- Symbolic Compilation
- Weighted Boolean Formula
- Weighted Model Count
- Probabilistic Circuit

Circuit compilation
Logic Circuit (BDD)

State of the art for discrete probabilistic program inference!
Conclusions

- Are we already in the age of computational abstractions?
- **Probabilistic circuits** for learning deep **tractable** probabilistic models
- **Probabilistic programs** as the new probabilistic knowledge representation language
- Two computational abstractions go hand in hand
Thanks

My students/postdoc who did the real work are graduating.

There are some awesome people on the academic job market!