AI can learn from data. But can it learn to reason?

Guy Van den Broeck

Oregon State University - Feb 3 2023
Outline

1. The paradox of learning to reason from data

2. Architectures for Learning and Reasoning

   a. Constrained language generation
   b. Constrained structured prediction
   c. Secret sauce: tractable circuits
Outline

1. The paradox of learning to reason from data
   deep learning

2. Architectures for Learning and Reasoning
   logical reasoning + deep learning
   
   a. Constrained language generation
   b. Constrained structured prediction
   c. Secret sauce: tractable circuits
Can Language Models Perform Logical Reasoning?

Language Models achieve high performance on various “reasoning” benchmarks in NLP.

It is unclear whether they solve the tasks following the rules of logical deduction.

**Language Models:**

\[
\text{input} \rightarrow ? \rightarrow \text{Carol is the grandmother of Justin.}
\]

**Logical Reasoning:**

\[
\text{input} \rightarrow \text{Justin in Kristin’s son; Carol is Kristin’s mother; } \rightarrow \text{Carol is Justin’s mother’s mother; if } X \text{ is } Y\text{’s mother’s mother then } X \text{ is } Y\text{’s grandmother } \rightarrow \text{Carol is the grandmother of Justin.}
\]
Generate textual train and test examples of the form:

Rules: If witty, then diplomatic. If careless and condemned and attractive, then blushing. If dishonest and inquisitive and average, then shy. If average, then stormy. If popular, then blushing. If talented, then hurt. If popular and attractive, then thoughtless. If blushing and shy and stormy, then inquisitive. If adorable, then popular. If cooperative and wrong and stormy, then thoughtless. If popular, then sensible. If cooperative, then wrong. If shy and cooperative, then witty. If polite and shy and thoughtless, then talented. If polite, then condemned. If polite and wrong, then inquisitive. If dishonest and inquisitive, then talented. If blushing and dishonest, then careless. If inquisitive and dishonest, then troubled. If blushing and stormy, then shy. If diplomatic and talented, then careless. If wrong and beautiful, then popular. If ugly and shy and beautiful, then stormy. If shy and inquisitive and attractive, then diplomatic. If witty and beautiful and frightened, then adorable. If diplomatic and cooperative, then sensible. If thoughtless and inquisitive, then diplomatic. If careless and dishonest and troubled, then cooperative. If hurt and witty and troubled, then dishonest. If scared and diplomatic and troubled, then average. If ugly and wrong and careless, then average. If dishonest and scared, then polite. If talented, then dishonest. If condemned, then wrong. If wrong and troubled and blushing, then scared. If attractive and condemned, then frightened. If hurt and condemned and shy, then witty. If cooperative, then attractive. If careless, then polite. If adorable and wrong and careless, then diplomatic. Facts: Alice sensible Alice condemned Alice thoughtless Alice polite Alice scared Alice average
Query: Alice is shy?
Problem Setting: SimpleLogic

The easiest of reasoning problems:

1. **Propositional logic** fragment
   a. bounded vocabulary & number of rules
   b. bounded reasoning depth (≤ 6)
   c. finite space (∼ 10^360)

2. **No language variance**: templated language

3. **Self-contained**
   No prior knowledge

4. **Purely symbolic** predicates
   No shortcuts from word meaning

5. **Tractable** logic (definite clauses)
   Can always be solved efficiently

Facts:
- Alice is *fast*.
- Alice is *normal*.

Rules:
- If Alice is *fast* and *smart*, then Alice is *bad*.
- If Alice is *normal*, then Alice is *smart*.
- If Alice is *normal* and *happy*, then Alice is *sad*.

Query 1: Alice is *bad*. [Answer: True]
Query 2: Alice is *sad*. [Answer: False]
Training a BERT model on SimpleLogic

(1) Randomly sample facts & rules.
Facts: B, C
Rules: A, B \to D, B \to E, B, C \to F.

(2) Compute the correct labels for all predicates given the facts and rules.

(1) Randomly assign labels to predicates.
True: B, C, E, F.
False: A, D.

Test accuracy for different reasoning depths

<table>
<thead>
<tr>
<th>Test</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>RP</td>
<td>99.9</td>
<td>99.8</td>
<td>99.7</td>
<td>99.3</td>
<td>98.3</td>
<td>97.5</td>
<td>95.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP</td>
<td>100.0</td>
<td>100.0</td>
<td>99.9</td>
<td>99.9</td>
<td>99.7</td>
<td>99.7</td>
<td>99.0</td>
</tr>
</tbody>
</table>

Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang and Guy Van den Broeck. On the Paradox of Learning to Reason from Data, 2022
Has BERT learned to reason from data?

1. Easiest of reasoning problems (no variance, self-contained, purely symbolic, tractable)
2. RP/LP data covers the whole problem space
3. The learned model has almost 100% test accuracy
4. There exist BERT parameters that compute the ground-truth reasoning function:

   **Theorem 1:** For a BERT model with $n$ layers and 12 attention heads, by construction, there exists a set of parameters such that the model can correctly solve any reasoning problem in SimpleLogic that requires at most $n - 2$ steps of reasoning.

Surely, under these conditions, BERT has learned the ground-truth reasoning function!
The Paradox of Learning to Reason from Data

1. If BERT has learned to reason, it should not exhibit such generalization failure.

2. If BERT has not learned to reason, it is baffling how it achieves near-perfect in-distribution test accuracy.

The BERT model trained on one distribution fails to generalize to the other distribution within the same problem space.
Why? Statistical Features

Monotonicity of entailment:
Any rules can be freely added to the hypothesis of any proven fact.

The more rules given, the more likely a predicate will be proved.

Pr(label = True | Rule # = x) should increase (roughly) monotonically with x.

(a) Statistics for examples generated by Rule-Priority (RP).
(b) Statistics for examples generated by Label-Priority (LP).
(c) Statistics for examples generated by uniform sampling;
BERT leverages statistical features to make predictions

RP_b downsamples from RP such that $\Pr(\text{label} = \text{True} \mid \text{rule#} = x) = 0.5$ for all $x$

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>RP</td>
<td>99.9</td>
<td>99.8</td>
</tr>
<tr>
<td>RP</td>
<td>99.0</td>
<td>99.3</td>
</tr>
</tbody>
</table>

1. Accuracy drop from RP to RP_b indicates that 
   the model is using rule# as a statistical feature to make predictions.

2. Potentially countless statistical features

3. Such features are **inherent to the reasoning problem**, cannot make data “clean”

Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang and Guy Van den Broeck. *On the Paradox of Learning to Reason from Data*, 2022
First Conclusion

Experiments unveil the fundamental difference between

1. learning to reason, and
2. learning to achieve high performance on benchmarks using statistical features.

Be careful deploying AI in applications where this difference matters.

Is more data going to solve this problem?
Outline

1. The paradox of learning to reason from data

2. Architectures for Learning and Reasoning

   a. Constrained language generation
   b. Constrained structured prediction
   c. Secret sauce: tractable circuits
Outline

1. The paradox of learning to reason from data
2. Architectures for Learning and Reasoning
   a. Constrained language generation
   b. Constrained structured prediction
   c. Secret sauce: tractable circuits
Is ChatGPT all you need?

- generate a sentence using the following keywords: fish watch kid swim tank

A kid watches a fish swim in the tank.

- generate a sentence using the following keywords: cook top pan stove vegetable

The cook used a pan on the stove top to prepare the vegetable.
Large language models are *not* all you need

*Fine-grained control (e.g., generation given lexical constraints) is challenging …*

Using the keywords "fish" "watch" "kid" "swim" "tank", generate a sentence with seven words:

Kid watches fish swim in tank. ✗
Large language models are not all you need

fine-grained control (e.g., generation given lexical constraints) is challenging …

generate a sentence using the following keywords and the keywords should appear in the order as listed: pan vegetable cook stove top

"She placed a pan of chopped vegetables on the stove top to **cook**."
Large language models are not all you need

*fine-grained control (e.g., generation given lexical constraints) is challenging …*

**H**

- generate a sentence with "pan" as the third word and "vegetable" as the fifth word.

---

**The chef used the pan to gently sauté the diced vegetable for their delicious stir-fry dish.**
Key Challenge: intractable conditioning

Given lexical constraint \( \alpha \):
“pan” “vegetable” “cook” “stove” “top” appears in the generated sentence

Computing \( \Pr(\text{sentence} \mid \alpha) \) is intractable for auto-regressive large language models (LLMs); in particular, computing \( \Pr(\text{next-token} \mid \alpha, \text{prefix}) \) is intractable.

Solution: language models that allow efficient conditioning

Tractable Probabilistic Models
A simple *Hidden Markov Model* architecture will suffice!

- 50k emission tokens $x$
- 4096 hidden states $z$
- trained as a *Probabilistic Circuit* in Juice.jl from LLM samples
- scaled up using *Latent Variable Distillation* [ICLR 2023]

---

**Step 1:** distill a TPM that *approximates* the distribution of an LLM.
Step 2: compute $\Pr(\text{next-token} \mid \text{prefix, } \alpha)$ via TPM

Dynamic programming algorithm in PyTorch that takes logical constraints $\alpha$. Can be quite complex: all inflections, all positions in seq, …

Step 3: control auto-regressive generation with the LLM.

Assume independence between fluency $\beta$ and constraint $\alpha$

$$
\Pr_{\text{TPM}}(x_{t+1} \mid x_{1:t}, \alpha, \beta) \\
\propto \Pr_{\text{TPM}}(\alpha \mid x_{1:t+1}, \beta) \cdot \Pr_{\text{TPM}}(x_{t+1} \mid x_{1:t}, \beta) \\
\propto \Pr_{\text{TPM}}(\alpha \mid x_{1:t+1}) \cdot \Pr_{\text{LM}}(x_{t+1} \mid x_{1:t})
$$

↳ sleight of hand
Controlling Language Generation via TPM

Lexical Constraint $\alpha$: the sentence contains keyword “winter”

Probabilistic Query: $\Pr(\text{next-token} \mid \alpha, \text{prefix} = \text{“the weather is”})$

- \text{intractable}
- \text{efficient}

Auto-regressive Language Model

- next-token: cold
  - $\Pr(\text{cold} \mid \text{prefix}) = 0.05$
- next-token: warm
  - $\Pr(\text{warm} \mid \text{prefix}) = 0.10$

Minimize KL-divergence

Tractable Probabilistic Model

- next-token: cold
  - $\Pr(\alpha \mid \text{next-token}, \text{prefix}) = 0.50$
- next-token: warm
  - $\Pr(\alpha \mid \text{next-token}, \text{prefix}) = 0.01$

- next-token: cold
  - $p(\text{cold} \mid \alpha, \text{prefix}) = 0.025$
- next-token: warm
  - $p(\text{warm} \mid \alpha, \text{prefix}) = 0.001$
### CommonGen: a challenging constrained generation task

<table>
<thead>
<tr>
<th>Method</th>
<th>Generation Quality</th>
<th>Constraint Satisfaction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ROUGE-L</td>
<td>BLEU-4</td>
</tr>
<tr>
<td><strong>Unsupervised</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InsNet (Lu et al., 2022a)</td>
<td>-</td>
<td>18.7</td>
</tr>
<tr>
<td>NeuroLogic (Lu et al., 2021)</td>
<td>-</td>
<td>24.7</td>
</tr>
<tr>
<td>A*esque (Lu et al., 2022b)</td>
<td>-</td>
<td>28.6</td>
</tr>
<tr>
<td>NADO (Meng et al., 2022)</td>
<td>-</td>
<td>26.2</td>
</tr>
<tr>
<td><strong>Ours</strong></td>
<td><strong>44.6</strong></td>
<td><strong>29.9</strong></td>
</tr>
<tr>
<td><strong>Supervised</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NeuroLogic (Lu et al., 2021)</td>
<td>-</td>
<td>26.7</td>
</tr>
<tr>
<td>A*esque (Lu et al., 2022b)</td>
<td>-</td>
<td>28.2</td>
</tr>
<tr>
<td>NADO (Meng et al., 2022)</td>
<td>30.8</td>
<td>-</td>
</tr>
<tr>
<td><strong>Ours</strong></td>
<td><strong>46.0</strong></td>
<td><strong>34.1</strong></td>
</tr>
</tbody>
</table>

State-of-the-art performance on the CommonGen dataset, beating baselines from various families of constrained generation techniques with a large margin. All baselines use GPT2-large as the base model.
Outline

1. The paradox of learning to reason from data

2. Architectures for Learning and Reasoning

   a. Constrained language generation
   b. Constrained structured prediction
   c. Secret sauce: tractable circuits
Warcraft Shortest Path

// for a 12 x 12 grid, $2^{144}$ states but only $10^{10}$ valid ones!

[Differentiation of Blackbox Combinatorial Solvers, Marin Vlastelica, Anselm Paulus, Vít Musil, Georg Martius, Michal Rolínek, 2019]
<table>
<thead>
<tr>
<th>Architecture</th>
<th>Exact Match</th>
<th>Hamming Score</th>
<th>Consistency</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-18+FIL</td>
<td>55.0</td>
<td>97.7</td>
<td>56.9</td>
</tr>
</tbody>
</table>

Is prediction the shortest path? This is the real task!
Are individual edge predictions correct?
Is output a path?

Declarative Knowledge of the Output

How is the output structured? Are all possible outputs valid?

How are the outputs related to each other?

Learning this from data is inefficient. Much easier to express this declaratively.
PyTorch Code

```python
for i in range(train_iters):
    ...
    py = model(x)
    ...
    loss = CrossEntropy(py,...)
```

1. Specify knowledge as a predicate

```python
def check(y):
    ...
    return isValid
```
PyTorch Code

```python
for i in range(train_iters):
    ...
    py = model(x)
    ...
    loss = CrossEntropy(py,...)
    loss += constraint_loss(check)(py)
```

1. Specify knowledge as a predicate
   ```python
def check(y):
    ...
    return isValid
```

2. Add as loss to training
   ```python
   loss += constraint_loss(check)(py)
   ```
PyTorch Code

for i in range(train_iters):
    ...
    py = model(x)
    ...
    loss = CrossEntropy(py,...)
    loss += constraint_loss(check)(py)

1. Specify knowledge as a predicate
   def check(y):
       ...
       return isValid

2. Add as loss to training
   loss += constraint_loss(check)

3. pylon derives the gradients (solves a combinatorial problem)
a) A network uncertain over both valid & invalid predictions

\[ L^s(\alpha, p) \propto -\log \sum_{x|\models \alpha} \prod_{i:x|\models X_i} p_i \prod_{i:x|\models \neg X_i} (1 - p_i) \]

Semantic Loss

Probability of satisfying constraint \( \alpha \) after sampling from neural net output layer \( p \)

In general: \#P-hard 😞

b) A network uncertain over both valid & invalid predictions

c) A network allocating most of its mass to models of constraint

Do this probabilistic-logical reasoning during learning in a computation graph
\[ \alpha: \ A \land B \Rightarrow C \]

Semantic Loss

Probability

\[ -\log( ) \]
Semantic Probabilistic Layers

- How to give a 100% guarantee that Boolean constraints will be satisfied?
- Bake the constraint into the neural network as a special layer

- Secret sauce is tractable circuits – computation graphs for reasoning
<table>
<thead>
<tr>
<th>Architecture</th>
<th>Exact Match</th>
<th>Hamming Score</th>
<th>Consistency</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-18+FIL</td>
<td>55.0</td>
<td>97.7</td>
<td>56.9</td>
</tr>
<tr>
<td>ResNet-18+$\mathcal{L}_{SL}$</td>
<td>59.4</td>
<td>97.7</td>
<td>61.2</td>
</tr>
<tr>
<td>ResNet-18+SPL</td>
<td>75.1</td>
<td>97.6</td>
<td>100.0</td>
</tr>
<tr>
<td>Overparam. SDD</td>
<td><strong>78.2</strong></td>
<td>96.3</td>
<td><strong>100.0</strong></td>
</tr>
</tbody>
</table>
Hierarchical Multi-Label Classification

“if the image is classified as a dog, it must also be classified as an animal”

“if the image is classified as an animal, it must be classified as either cat or dog”

<table>
<thead>
<tr>
<th>DATASET</th>
<th>EXACT MATCH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HMCNN</td>
</tr>
<tr>
<td>CELL_CYCLE</td>
<td>3.05 ± 0.11</td>
</tr>
<tr>
<td>DERISI</td>
<td>1.39 ± 0.47</td>
</tr>
<tr>
<td>EISEN</td>
<td>5.40 ± 0.15</td>
</tr>
<tr>
<td>EXPR</td>
<td>4.20 ± 0.21</td>
</tr>
<tr>
<td>GASCH1</td>
<td>3.48 ± 0.96</td>
</tr>
<tr>
<td>GASCH2</td>
<td>3.11 ± 0.08</td>
</tr>
<tr>
<td>SEQ</td>
<td>5.24 ± 0.27</td>
</tr>
<tr>
<td>SPO</td>
<td>1.97 ± 0.06</td>
</tr>
<tr>
<td>DIATOMS</td>
<td>48.21 ± 0.57</td>
</tr>
<tr>
<td>ENRON</td>
<td>5.97 ± 0.56</td>
</tr>
<tr>
<td>IMCLEF07A</td>
<td>79.75 ± 0.38</td>
</tr>
<tr>
<td>IMCLEF07D</td>
<td>76.47 ± 0.35</td>
</tr>
</tbody>
</table>
**Example.**

**Learning to Explain (L2X)**

<table>
<thead>
<tr>
<th>Taste Score</th>
<th>Key Words (k = 10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7</td>
<td>a lite bodied beer with a pleasant taste. was like a reddish color. a little like wood and caramel with a hop finish. has a sort of fruity flavor like grapes or cherry that is sort of buried in there. mouth feel was lite, sort of bubbly. not hard to down, though a bit harder then one would expect given the taste.</td>
</tr>
</tbody>
</table>
**SIMPLE**: Gradient Estimator for $k$-Subset Sampling

We achieve **lower bias and variance** by exact, discrete samples and exact derivative of conditional marginals.

Experiment: Learn to Explain (L2X)

<table>
<thead>
<tr>
<th>Taste Score</th>
<th>Key Words ((k = 10))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7</td>
<td>a lite bodied beer with a pleasant taste, was like a reddish color. a little like wood and caramel with a hop finish. has a sort of fruity flavor like grapes or cherry that is sort of buried in there. mouth feel was lite, sort of bubbly. not hard to down, though a bit harder then one would expect given the taste.</td>
</tr>
</tbody>
</table>

Results for three aspects with \(k = 10\)

<table>
<thead>
<tr>
<th>Method</th>
<th>Test MSE</th>
<th>Precision</th>
<th>Test MSE</th>
<th>Precision</th>
<th>Test MSE</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMPLE (Ours)</td>
<td>2.35 ± 0.28</td>
<td>66.81 ± 7.56</td>
<td>2.68 ± 0.06</td>
<td>44.78 ± 2.75</td>
<td>2.11 ± 0.02</td>
<td>42.31 ± 0.61</td>
</tr>
<tr>
<td>L2X ((t = 0.1))</td>
<td>10.70 ± 4.82</td>
<td>30.02 ± 15.82</td>
<td>6.70 ± 0.63</td>
<td>50.39 ± 13.58</td>
<td>6.92 ± 1.61</td>
<td>32.23 ± 4.92</td>
</tr>
<tr>
<td>SoftSub ((t = 0.5))</td>
<td>2.48 ± 0.10</td>
<td>52.86 ± 7.08</td>
<td>2.94 ± 0.08</td>
<td>39.17 ± 3.17</td>
<td>2.18 ± 0.10</td>
<td>41.98 ± 1.42</td>
</tr>
<tr>
<td>I-MLE ((\tau = 30))</td>
<td>2.51 ± 0.05</td>
<td>65.47 ± 4.95</td>
<td>2.96 ± 0.04</td>
<td>40.73 ± 3.15</td>
<td>2.38 ± 0.04</td>
<td>41.38 ± 1.55</td>
</tr>
</tbody>
</table>

Results for aspect Aroma, for \(k\) in \{5, 10, 15\}

<table>
<thead>
<tr>
<th>Method</th>
<th>(k = 5)</th>
<th>(k = 10)</th>
<th>(k = 15)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Test MSE</td>
<td>Precision</td>
<td>Test MSE</td>
</tr>
<tr>
<td>SIMPLE (Ours)</td>
<td>2.27 ± 0.05</td>
<td>57.30 ± 3.04</td>
<td>2.23 ± 0.03</td>
</tr>
<tr>
<td>L2X ((t = 0.1))</td>
<td>5.75 ± 0.30</td>
<td>33.63 ± 6.91</td>
<td>6.68 ± 1.08</td>
</tr>
<tr>
<td>SoftSub ((t = 0.5))</td>
<td>2.57 ± 0.12</td>
<td>54.06 ± 6.29</td>
<td>2.67 ± 0.14</td>
</tr>
<tr>
<td>I-MLE ((\tau = 30))</td>
<td>2.62 ± 0.05</td>
<td>54.76 ± 2.50</td>
<td>2.71 ± 0.10</td>
</tr>
</tbody>
</table>

Kareem Ahmed, Zhe Zeng, Mathias Niepert, Guy Van den Broeck. SIMPLE: A Gradient Estimator for k-Subset Sampling. ICLR 2023
Outline

1. The paradox of learning to reason from data
   
2. Architectures for Learning and Reasoning
   logical reasoning + deep learning
   
   a. Constrained language generation
   b. Constrained structured prediction
   c. Secret sauce: tractable circuits
Probabilistic circuits

*computational graphs* that recursively define distributions

\[ x \xrightarrow{\bigwedge} X \xrightarrow{\bigwedge} p_X(x) \]

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

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

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

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

A product node is *decomposable* if its children depend on disjoint sets of variables.

---

*smooth circuit*

*decomposable circuit*
**Smoothness + decomposability = tractable MAR**

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

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

\[\Rightarrow \text{integrals are “pushed down” to children}\]
If \( p(x, y, z) = p(x)p(y)p(z) \), (decomposability):

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

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

Forward pass evaluation for MAR

⇒ *linear in circuit size!*

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

- leafs over $X_1$ and $X_3$ output $Z_i = \int p(x_i)dx_i$
  ⇒ *for normalized leaf distributions: \textbf{1.0}*

- leafs over $X_2$ and $X_4$ output \textbf{EVI}

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

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

E.g. to compute \( p(x_2, x_4) \):
- 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 \text{EVI}
- feedforward evaluation (bottom-up)
Learn more about probabilistic circuits?

**Tutorial (3h)**

Probabilistic Circuits

Inference
Representations
Learning Theory

Antonio Vergari
University of California, Los Angeles

YooJung Choi
University of California, Los Angeles

Robert Peharz
TU Eindhoven

Guy Van den Broeck
University of California, Los Angeles

September 14th, 2020 - Ghent, Belgium - ECML-PKDD 2020

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

Outline

1. The paradox of learning to reason from data
   
2. Architectures for Learning and Reasoning
   
   a. Constrained language generation
   b. Constrained structured prediction
   c. Secret sauce: tractable circuits
Thanks

This was the work of many wonderful students/postdocs/collaborators!

References: http://starai.cs.ucla.edu/publications/