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

Guy Van den Broeck

The 18th Reasoning Web Summer School - Sep 28 2022
Integrate reasoning into modern deep learning algorithms
Outline

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

2. Tractable deep generative models
   probabilistic reasoning + deep learning

3. Learning with symbolic knowledge
   logical reasoning + deep learning
Outline

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

2. Tractable deep generative models
   probabilistic reasoning + deep learning

3. Learning with symbolic knowledge
   logical reasoning + deep learning
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:**

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

**Reasoning:**

\[ \text{input} \rightarrow \text{Justin in Kristin’s son; Carol is Kristin’s mother; } \rightarrow \text{Carol is Justin’s mother’s mother; if } X \text{ is } Y \text{’s mother’s mother then } X \text{ is } Y \text{’s grandmother } \rightarrow \text{Carol is the grandmother of Justin.} \]
Problem Setting: SimpleLogic

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 ($\leq 6$)
   c. finite space ($\approx 10^{360}$)

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]

LMs: BERT, T5

True or False
Training a BERT model on SimpleLogic

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

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

Rule-Priority

D
E
F
A
B
C

Label-Priority

D
E
F
A
B
C

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

(2) Set B, C (randomly chosen among B, C, E, F) as facts and sample rules (randomly) consistent with the label assignments.

Test accuracy for different reasoning depths

<table>
<thead>
<tr>
<th></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></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.

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<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>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>
<td></td>
</tr>
<tr>
<td></td>
<td>LP</td>
<td>99.8</td>
<td>99.8</td>
<td>99.3</td>
<td>96.0</td>
<td>90.4</td>
<td>75.0</td>
<td>57.3</td>
<td></td>
</tr>
<tr>
<td>LP</td>
<td>RP</td>
<td>97.3</td>
<td>66.9</td>
<td>53.0</td>
<td>54.2</td>
<td>59.5</td>
<td>65.6</td>
<td>69.2</td>
<td></td>
</tr>
<tr>
<td></td>
<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>
<td></td>
</tr>
</tbody>
</table>

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

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
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

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

2. Though removing one statistical feature from training data can help with model generalization, there are potentially countless statistical features and it is computationally infeasible to jointly remove them.

<table>
<thead>
<tr>
<th>Train</th>
<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>RP_b</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>
<tr>
<td>RP</td>
<td>RP_b</td>
<td>99.0</td>
<td>99.3</td>
<td>98.5</td>
<td>97.5</td>
<td>96.7</td>
<td>93.5</td>
<td>88.3</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
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.
Outline

1. The paradox of learning to reason from data

2. Tractable deep generative models

3. Learning with symbolic knowledge
Intractable and tractable models
"Every talk needs a joke and a literature overview slide, not necessarily distinct"
- after Ron Graham
Probabilistic circuits

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

\[
p(X_1) = w_1 p_1(X_1) + w_2 p_2(X_1)
\]
\[\Rightarrow \text{mixtures}\]

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

*computational graphs* that recursively define distributions

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

\[ p(X_1, X_2) = p(X_1) \cdot p(X_2) \]

⇒ *mixtures*

⇒ *factorizations*
Tractable Probabilistic Inference

A class of queries $Q$ is tractable on a family of probabilistic models $M$ iff for any query $q \in Q$ and model $m \in M$ exactly computing $q(m)$ runs in time $O(poly(|m|))$.

$\Rightarrow$ often $poly$ will in fact be linear!

$\Rightarrow$ Note: if $M$ is compact in the number of random variables $X$, that is, $|m| \in O(poly(|X|))$, then query time is $O(poly(|X|))$. 
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
Tractable marginals

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*
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]
Smoothness + decomposability = tractable MAR

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

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

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

Forward pass evaluation for MAR

⇒ linear in circuit size!

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

1. leaves over $X_1$ and $X_3$ output $Z_i = \int p(x_i)dx_i$

⇒ for normalized leaf distributions: 1.0

2. leaves over $X_2$ and $X_4$ output EVI

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

Forward pass evaluation for MAR

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

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

- Leaf over \( X_1 \) and \( X_3 \) output \( Z_i = \int p(x_i)dx_i \)

\[ \Rightarrow \text{for normalized leaf distributions: } \mathbf{1.0} \]

- Leaf 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
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]
Where do architectures come from?
Where do architectures come from?

Where do architectures come from?

Gens and Domingos, “Learning the Structure of Sum-Product Networks”, 2013
Where do architectures come from?

[Diagram showing a tree structure with nodes labeled A, B, C, D, E, F, and edges indicating dependencies.]

[Rahman et al. 2014]
Where do architectures come from?

\[ S = W \text{ vec}(P) \]

\[ P = N \otimes N' \]
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
From BN trees to circuits
via compilation

...compile a leaf CPT

\[ p(A|C = 0) \]

\[ \begin{aligned}
  &+ \\
  &.3 \quad .7 \\
  &A = 0 \quad A = 1
\end{aligned} \]
From BN trees to circuits

via compilation

...compile a leaf CPT...for all leaves...

\[ p(A|C') \]

\[
\begin{align*}
A = 0 & \quad A = 1 \\
\end{align*}
\]

\[ p(B|C) \]

\[
\begin{align*}
B = 0 & \quad B = 1 \\
\end{align*}
\]
From BN trees to circuits
via compilation

...and recurse over parents...
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**

Data → Encode → **10110** → Decode → Reconstructed data

- **Expressive probabilistic model** $p(x)$
  - Determines the theoretical limit of compression rate
- Efficient coding algorithm
  - How close we can approach the theoretical limit

A Typical Streaming Code – Arithmetic Coding

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

Need to compute

\[
\begin{align*}
& p(X_1 < x_1) \\
& p(X_1 \leq x_1) \\
& p(X_2 < x_2 | x_1) \\
& p(X_2 \leq x_2 | x_1) \\
& p(X_3 < x_3 | x_1, x_2) \\
& p(X_3 \leq x_3 | x_1, x_2) \\
& \vdots
\end{align*}
\]
Lossless Neural Compression with Probabilistic Circuits

Probabilistic Circuits
- Expressive
- Fast

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

Arithmetic Coding:

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

...
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</td>
<td>(1.98)</td>
</tr>
<tr>
<td>FashionMNIST</td>
<td>3.37 (3.34)</td>
<td>3.50 (3.47)</td>
<td><strong>3.35</strong> (3.28)</td>
<td>3.69 (3.66)</td>
<td>3.93</td>
<td>4.62</td>
<td>(3.72)</td>
</tr>
<tr>
<td>EMNIST (Letter)</td>
<td><strong>1.84</strong> (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</td>
<td>(3.12)</td>
</tr>
<tr>
<td>EMNIST (ByClass)</td>
<td><strong>1.89</strong> (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</td>
<td>(3.14)</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

Mehhua Dang, Anji Liu, Xinzhu Wei, Sriram Sankararaman, and Guy Van den Broeck, Tractable and expressive generative models of genetic variation data, RECOMB 2022
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>
Expressive models without compromises
We cannot decompose bottom-up a MAP query:

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

since for a sum node we are marginalizing out a latent variable

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

$\Rightarrow$ MAP for latent variable models is intractable [Conaty et al. 2017]
Determinism

A sum node is \textit{deterministic} if only one of its children outputs non-zero for any input

\[
\Rightarrow \text{allows tractable MAP inference} \\
\arg\max_x p(x)
\]

\text{deterministic circuit}

Darwiche and Marquis, “A Knowledge Compilation Map”, 2002
\[
\text{Determinism} + \text{decomposability} = \text{tractable MAP}
\]

If \( p(q, e) = \sum_i w_i p_i(q, e) = \max_i w_i p_i(q, e) \),

(deterministic sum node):

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

\[\Rightarrow\] one non-zero child term, thus sum is max
Determinism + decomposability = tractable MAP

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):

\[
\max_q p(q \mid e) = \max_q p(q, e) \\
= \max_{q_x, q_y} p(q_x, e_x, q_y, e_y) \\
= \max_{q_x} p(q_x, e_x) \cdot \max_{q_y} p(q_y, e_y)
\]

\( \Rightarrow \) solving optimization independently
Determinism + decomposability = tractable MAP

Evaluating the circuit twice: bottom-up and top-down \(\Rightarrow\) still linear in circuit size!

E.g., for \(\text{argmax}_{x_1, x_3} p(x_1, x_3 \mid x_2, x_4)\):

1. turn sum into max nodes and distributions into max distributions
2. evaluate \(p(x_2, x_4)\) bottom-up
3. retrieve max activations top-down
4. compute MAP states for \(X_1\) and \(X_3\) at leaves
Queries as pipelines: KLD

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

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

\[ p \quad \xrightarrow{\text{log}} \quad q \]

\[ \Rightarrow \text{we can reuse the operations!} \]
<table>
<thead>
<tr>
<th>Operation</th>
<th>( \log(p) )</th>
<th>Input conditions</th>
<th>Output conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LOG</strong></td>
<td></td>
<td>Sm, Dec, Det</td>
<td>Sm, Dec</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><strong>SUM</strong> $\theta_1 p + \theta_2 q$</td>
<td>(+Cmp)</td>
<td>(+SD)</td>
<td>NP-hard for Det output</td>
</tr>
<tr>
<td><strong>PRODUCT</strong> $p \cdot q$</td>
<td>Cmp (+Det, +SD)</td>
<td>Dec (+Det, +SD)</td>
<td>#P-hard w/o Cmp</td>
</tr>
<tr>
<td><strong>POWER</strong> $p^n, n \in \mathbb{N}$</td>
<td>SD (+Det)</td>
<td>SD (+Det)</td>
<td>#P-hard w/o SD</td>
</tr>
<tr>
<td>$p^\alpha, \alpha \in \mathbb{R}$</td>
<td>Sm, Dec, Det (+SD)</td>
<td>Sm, Dec, Det (+SD)</td>
<td>#P-hard w/o Det</td>
</tr>
<tr>
<td><strong>QUOTIENT</strong> $p/q$</td>
<td>Cmp; $q$ Det (+$p$ Det, +SD)</td>
<td>Dec (+$Det$, +SD)</td>
<td>#P-hard w/o Det</td>
</tr>
<tr>
<td><strong>LOG</strong> $\log(p)$</td>
<td>Sm, Dec, Det</td>
<td>Sm, Dec</td>
<td>#P-hard w/o Det</td>
</tr>
<tr>
<td><strong>EXP</strong> $\exp(p)$</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><strong>CROSS ENTROPY</strong></td>
<td>Cmp, q Det</td>
<td>#P-hard w/o Det</td>
</tr>
<tr>
<td><strong>SHANNON ENTROPY</strong></td>
<td>Sm, Dec, Det</td>
<td>coNP-hard w/o Det</td>
</tr>
<tr>
<td><strong>RéNYI ENTROPY</strong></td>
<td>SD</td>
<td>#P-hard w/o SD</td>
</tr>
<tr>
<td><strong>MUTUAL INFORMATION</strong></td>
<td>Sm, Dec, Det*</td>
<td>coNP-hard w/o SD</td>
</tr>
<tr>
<td><strong>KULLBACK-LEIBLER DIV.</strong></td>
<td>Cmp, Det</td>
<td>#P-hard w/o Det</td>
</tr>
<tr>
<td><strong>RéNYI’S ALPHA DIV.</strong></td>
<td>Cmp, q Det</td>
<td>#P-hard w/o Det</td>
</tr>
<tr>
<td><strong>ITAKURA-SAITO DIV.</strong></td>
<td>Cmp, Det</td>
<td>#P-hard w/o Det</td>
</tr>
<tr>
<td><strong>CAUCHY-SCHWARZ DIV.</strong></td>
<td>Cmp</td>
<td>#P-hard w/o Cmp</td>
</tr>
<tr>
<td><strong>SQUARED LOSS</strong></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: $\arg\max_q p(q|e)$

$\Rightarrow$ i.e. MAP of a marginal distribution on $Q$

\[\text{NP}^{PP}\text{-complete} \text{ for PGMs}\]

\[\text{NP-hard} \text{ even for PCs tractable for marginals, MAP & entropy}\]
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 \textit{linear time}
Iterative MMAP solver

Prune edges

Tighten bounds

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

$$E_{x \sim p, x' \sim q}[k(x, x')]$$

- 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

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

importance weights

expected kernel matrix

tractability is a spectrum
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

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

Probabilistic Circuits

- Positive Dependence
- Fully Factorized

Determinantal Point Processes

We cannot tractably represent DPPs with subclasses of PCs

PSDDs

More Tractable

Deterministic and Decomposable PCs

Deterministic PCs with no negative parameters

Fewer Constraints

Deterministic PCs with negative parameters

Decomposable PCs with no negative parameters (SPNs)

We don’t know

Decomposable PCs with negative parameters

No

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>$\Pr_\beta$</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><strong>1</strong></td>
<td><strong>1</strong></td>
<td><strong>1</strong></td>
<td><strong>0.16</strong></td>
</tr>
</tbody>
</table>

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

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

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

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

### Graphical Model

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

\(\alpha_k\) gives the answer

Example

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

---

<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>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>
## Experiment Results: Amazon Baby Registries

<table>
<thead>
<tr>
<th>Category</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>&lt;sup&gt;↑&lt;/sup&gt;</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>&lt;sup&gt;↑&lt;/sup&gt;</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>&lt;sup&gt;↑&lt;/sup&gt;</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>&lt;sup&gt;↑&lt;/sup&gt;</td>
</tr>
<tr>
<td>feeding</td>
<td>-11.86</td>
<td>-11.42</td>
<td>-11.27</td>
<td>-11.30</td>
<td><strong>-11.17</strong>&lt;sup&gt;↑&lt;/sup&gt;</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>&lt;sup&gt;↑&lt;/sup&gt;</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&lt;sup&gt;°&lt;/sup&gt;</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>&lt;sup&gt;↑&lt;/sup&gt;</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&lt;sup&gt;↑&lt;/sup&gt;</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&lt;sup&gt;°&lt;/sup&gt;</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>&lt;sup&gt;↑&lt;/sup&gt;</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>&lt;sup&gt;↑&lt;/sup&gt;</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&lt;sup&gt;↑&lt;/sup&gt;</td>
</tr>
</tbody>
</table>

SimplePGC achieves SOTA result on 11/15 datasets
Learn more about probabilistic circuits?

**Tutorial (3h)**

[Image 1]

**Overview Paper (80p)**

Proportional 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  
2 Probabilistic Inference: Models, Queries, and Tractability  
   2.1 Probabilistic Models  
   2.2 Probabilistic Queries  
   2.3 Tractable Probabilistic Inference  
   2.4 Properties of Tractable Probabilistic Models

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

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>3.66</td>
</tr>
<tr>
<td>EMNIST (Letter split)</td>
<td>1.80</td>
<td>1.95</td>
<td>1.84</td>
<td>2.26</td>
</tr>
<tr>
<td>EMNIST (ByClass split)</td>
<td>1.85</td>
<td>1.98</td>
<td>1.87</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:
Outline

1. The paradox of learning to reason from data
   
2. Tractable deep generative models
   
3. Learning with symbolic knowledge
The AI Dilemma

Pure (Logic) Reasoning

- Slow thinking: deliberative, cognitive, model-based, extrapolation
- Amazing achievements until this day

"Pure logic is brittle"
noise, uncertainty, incomplete knowledge, ...

Pure Learning
The AI Dilemma

- Fast thinking: instinctive, perceptive, model-free, interpolation
- Amazing achievements recently
- “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 fails to incorporate a sensible model of the world
Knowledge in Vision, Robotics, NLP, Activity Recognition

People appear at most once in a frame

Rigid objects don’t overlap

At least one verb in each sentence.

If X and Y are married, then they are people.

Cut the orange before squeezing the orange

Predict Loan Amount

Neural Network Model: *Increasing income can decrease the approved loan amount*

Monotonicity (Prior Knowledge):  
*Increasing income should increase the approved loan amount*
Motivation: Deep Learning

Motivation: Deep Learning

DeepMind’s latest technique uses external memory to solve tasks that require logic and reasoning — a step toward more human-like AI.

... but ...

optimal planner recalculating a shortest path to the end node. To ensure that the network always moved to a valid node, the output distribution was renormalized over the set of possible triples outgoing from the current node. The performance it also received input triples during the answer phase, indicating the actions chosen on the previous time-step. This makes the problem a ‘structured prediction’

Warcraft Shortest Path

Predicting the minimum-cost path

[Differentiation of Blackbox Combinatorial Solvers, Marin Vlastelica, Anselm Paulus, Vít Musil, Georg Martius, Michal Rolínek, 2019]
Knowledge vs. Data

• Where did the world knowledge go?
  – Python scripts
    • Decode/encode/search cleverly
    • Fix inconsistent beliefs
  – Rule-based decision systems
  – Dataset design
  – “a big hack” (with author’s permission)

• In some sense we went backwards
  Less principled, scientific, and intellectually satisfying ways of incorporating knowledge
## Warcraft min-cost simple-path prediction results

<table>
<thead>
<tr>
<th>Test accuracy %</th>
<th>Coherent</th>
<th>Incoherent</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-18</td>
<td>44.8</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?*

A PyTorch Framework for Learning with Constraints

Kareem Ahmed    Tao Li      Thy Ton      Quan Guo,
Kai-Wei Chang    Parisa Kordjamshidi   Vivek Srikumar
Guy Van den Broeck   Sameer Singh

http://pylon-lib.github.io
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
Library that extends PyTorch to allow injection of declarative knowledge

- **Easy to Express Knowledge**: users write arbitrary constraints on the output
- **Integrates with PyTorch**: minimal change to existing code
- **Efficient Training**: compiles into loss that can be efficiently optimized
  - Exact semantic loss (see later)
  - Monte-carlo estimate of loss
  - T-norm approximation
  - *your solver?*
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)
   ```
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)
   ```

3. `pylon` derives the gradients (solves a combinatorial problem)
### Warcraft min-cost simple-path prediction results

<table>
<thead>
<tr>
<th>Test accuracy %</th>
<th>Coherent</th>
<th>Incoherent</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-18</td>
<td>44.8</td>
<td>97.7</td>
<td>56.9</td>
</tr>
<tr>
<td>+ Semantic loss</td>
<td>50.9</td>
<td>97.7</td>
<td>67.4</td>
</tr>
</tbody>
</table>
Q: How close is output $p$ to satisfying constraint $\alpha$?
A: Semantic loss function $L(\alpha, p)$

- Axioms, for example:
  - If $\alpha$ constrains to one label, $L(\alpha, p)$ is cross-entropy
  - If $\alpha$ implies $\beta$ then $L(\alpha, p) \geq L(\beta, p)$ \textit{(\alpha more strict)}

- Implied Properties:
  - If $\alpha$ is equivalent to $\beta$ then $L(\alpha, p) = L(\beta, p)$
  - If $p$ is Boolean and satisfies $\alpha$ then $L(\alpha, p) = 0$
Axioms imply unique semantic loss:

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

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

In general: \#P-hard 😞

Do this probabilistic-logical reasoning during learning in a computation graph
Circuits = Computation Graphs

- Logical circuits that can count solutions (#SAT) also compute semantic loss efficiently in size of circuit

\[
L(\alpha, p) = L(\alpha, p) = - \log(\text{Pr}(\text{false}))
\]

- Compilation into circuit by SAT solvers (once)
- Add circuit to neural network output in pytorch/tensorflow/...
\[ \alpha: A \land B \Rightarrow C \]

- \( \log(\ ) \)  

Semantic Loss

Probability

\( p \)
a) A network uncertain over both valid & invalid predictions

b) A network allocating most of its mass to an invalid prediction.

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

d) A network allocating most of mass to one model of formula
Two complementary neuro-symbolic losses

$$P(Y|x) \quad \text{Semantic Loss} \quad P(\alpha|x) \uparrow: - \log P(\alpha|x) \downarrow$$

$$-\mathbb{E}_{P(Y|x,\alpha)}[\log P(Y|x,\alpha)]$$
Warcraft min-cost simple-path prediction results

<table>
<thead>
<tr>
<th>Test accuracy %</th>
<th>Coherent</th>
<th>Incoherent</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-18</td>
<td>44.8</td>
<td>97.7</td>
<td>56.9</td>
</tr>
<tr>
<td>Semantic loss</td>
<td>50.9</td>
<td>97.7</td>
<td>67.4</td>
</tr>
<tr>
<td>+ Full Entropy</td>
<td>51.5</td>
<td>97.6</td>
<td>67.7</td>
</tr>
<tr>
<td>+ NeSy Entropy</td>
<td>55.0</td>
<td>97.9</td>
<td>69.8</td>
</tr>
</tbody>
</table>
Joint entity-relation extraction in natural language processing

<table>
<thead>
<tr>
<th>#</th>
<th>3</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>25</th>
<th>50</th>
<th>75</th>
</tr>
</thead>
</table>
| ACE05
| Baseline | 4.92 ± 1.12 | 7.24 ± 1.75 | 13.66 ± 0.18 | 15.07 ± 1.79 | 21.65 ± 3.41 | 28.96 ± 0.98 | 33.02 ± 1.17 |
| Self-training | 7.72 ± 1.21 | 12.83 ± 2.97 | 16.22 ± 3.08 | 17.55 ± 1.41 | 27.00 ± 3.66 | 32.90 ± 1.71 | 37.15 ± 1.42 |
| Product t-norm | 8.89 ± 5.09 | 14.52 ± 2.13 | 19.22 ± 5.81 | 21.80 ± 7.67 | 30.15 ± 1.01 | 34.12 ± 2.75 | 37.35 ± 2.53 |
| Semantic Loss | 12.00 ± 3.81 | 14.92 ± 3.14 | 22.23 ± 3.64 | 27.35 ± 3.10 | 30.78 ± 0.68 | 36.76 ± 1.40 | 38.49 ± 1.74 |
| + Full Entropy | 14.80 ± 3.70 | 15.78 ± 1.90 | 23.34 ± 4.07 | 28.09 ± 1.46 | 31.13 ± 2.26 | 36.05 ± 1.00 | 39.39 ± 1.21 |
| + NeSy Entropy | 14.72 ± 1.57 | 18.38 ± 2.50 | 26.41 ± 0.49 | 31.17 ± 1.68 | 35.85 ± 0.75 | 37.62 ± 2.17 | 41.28 ± 0.46 |
| SciERC
| Baseline | 2.71 ± 1.10 | 2.94 ± 1.00 | 3.49 ± 1.80 | 3.56 ± 1.10 | 8.83 ± 1.00 | 12.32 ± 3.00 | 12.49 ± 2.60 |
| Self-training | 3.56 ± 1.40 | 3.04 ± 0.90 | 4.14 ± 2.60 | 3.73 ± 1.10 | 9.44 ± 3.80 | 14.82 ± 1.20 | 13.79 ± 3.90 |
| Product t-norm | 6.50 ± 2.00 | 8.86 ± 1.20 | 10.92 ± 1.60 | 13.38 ± 0.70 | 13.83 ± 2.90 | 19.20 ± 1.70 | 19.54 ± 1.70 |
| Semantic Loss | 6.47 ± 1.02 | 9.31 ± 0.76 | 11.50 ± 1.53 | 12.97 ± 2.86 | 14.07 ± 2.33 | 20.47 ± 2.50 | 23.72 ± 0.38 |
| + Full Entropy | 6.26 ± 1.21 | 8.49 ± 0.85 | 11.12 ± 1.22 | 14.10 ± 2.79 | 17.25 ± 2.75 | 22.42 ± 0.43 | 24.37 ± 1.62 |
| + NeSy Entropy | 6.19 ± 2.40 | 8.11 ± 3.66 | 13.17 ± 1.08 | 15.47 ± 2.19 | 17.45 ± 1.52 | 22.14 ± 1.46 | 25.11 ± 1.03 |

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 again tractable circuits – computation graphs for reasoning
Warcraft Shortest Path

GROUNDS TRUTH  RESNET-18  SEMANTIC LOSS  SPL (ours)
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”
Neuro-Symbolic Learning Settings

Learn

1. neural network given symbols and constraints and data

2. neural network and constraints given symbols and data

3. neural network and constraints and symbols given data

Everyone is working on 1. Ongoing work on 2.
Neuro-Symbolic Joint Training

Learn invariant features using neural networks. Learn logic to tie it all together.

Ask Yitao Liang, Anji Liu
Neuro-Symbolic Joint Training

<table>
<thead>
<tr>
<th>Model</th>
<th>Multi-digit addition [test seq length + train/test img]</th>
<th>Tower of Hanoi</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5 w/ test</td>
<td>10 w/ test</td>
</tr>
<tr>
<td>DeepProbLog†</td>
<td>88.30</td>
<td>77.46</td>
</tr>
<tr>
<td>LSTM</td>
<td>81.40</td>
<td>56.97</td>
</tr>
<tr>
<td>DNC</td>
<td>81.49</td>
<td>59.64</td>
</tr>
<tr>
<td>NToC(ours)</td>
<td><strong>89.82</strong></td>
<td><strong>77.97</strong></td>
</tr>
</tbody>
</table>

Learn invariant features using neural networks. Learn logic to tie it all together.

Ask Yitao Liang, Anji Liu
Predict Loan Amount

Neural Network Model: Increasing income can decrease the approved loan amount

Monotonicity (Prior Knowledge): Increasing income should increase the approved loan amount
Counterexamples

\[ \exists x, y \ x \leq y \implies f(x) > f(y) \]

Computed using SMT(LRA) logical reasoning solver

Maximal counterexamples (largest violation) using OMT

Counterexample-Guided Predictions

Monotonic Envelope:
- Replace each prediction by its maximal counterexample
- Envelope construction is online (during prediction)
- Guarantees monotonic predictions for any ReLU neural net
- Works for high-dimensional input
- Works for multiple monotonic features

## Monotonic Envelope: Performance

### Guaranteed monotonicity at little to no cost

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Feature</th>
<th>( \text{NN}_b )</th>
<th>Envelope</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Auto-MPG</strong></td>
<td>Weight</td>
<td>9.33±3.22</td>
<td><strong>9.19±3.41</strong></td>
</tr>
<tr>
<td></td>
<td>Displ.</td>
<td>9.33±3.22</td>
<td>9.63±2.61</td>
</tr>
<tr>
<td></td>
<td>W,D</td>
<td>9.33±3.22</td>
<td>9.63±2.61</td>
</tr>
<tr>
<td></td>
<td>W,D,HP</td>
<td>9.33±3.22</td>
<td>9.63±2.61</td>
</tr>
<tr>
<td><strong>Boston</strong></td>
<td>Rooms</td>
<td>14.37±2.4</td>
<td><strong>14.19±2.28</strong></td>
</tr>
<tr>
<td></td>
<td>Crime</td>
<td>14.37±2.4</td>
<td><strong>14.02±2.17</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Feature</th>
<th>( \text{NN}_b )</th>
<th>Envelope</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Heart</strong></td>
<td>Trestbps</td>
<td>0.85±0.04</td>
<td>0.85±0.04</td>
</tr>
<tr>
<td></td>
<td>Chol.</td>
<td>0.85±0.04</td>
<td>0.85±0.05</td>
</tr>
<tr>
<td></td>
<td>T,C</td>
<td>0.85±0.04</td>
<td>0.85±0.05</td>
</tr>
<tr>
<td><strong>Adult</strong></td>
<td>Cap. Gain</td>
<td>0.84</td>
<td>0.84</td>
</tr>
<tr>
<td></td>
<td>Hours</td>
<td>0.84</td>
<td>0.84</td>
</tr>
</tbody>
</table>

Counterexample-Guided Learning

How to use monotonicity to improve model quality?
“Monotonicity as inductive bias”

Counterexample-Guided Learning: Performance

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Feature</th>
<th>$\text{NN}_b$</th>
<th>$\text{CGL}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto-MPG</td>
<td>Weight</td>
<td>9.33±3.22</td>
<td>9.04±2.76</td>
</tr>
<tr>
<td></td>
<td>Displ.</td>
<td>9.33±3.22</td>
<td>9.08±2.87</td>
</tr>
<tr>
<td></td>
<td>W,D</td>
<td>9.33±3.22</td>
<td>8.86±2.67</td>
</tr>
<tr>
<td></td>
<td>W,D,HP</td>
<td>9.33±3.22</td>
<td>8.63±2.21</td>
</tr>
<tr>
<td>Boston</td>
<td>Rooms</td>
<td>14.37±2.4</td>
<td>12.24±2.87</td>
</tr>
<tr>
<td></td>
<td>Crime</td>
<td>14.37±2.4</td>
<td>11.66±2.89</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Feature</th>
<th>$\text{NN}_b$</th>
<th>$\text{CGL}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heart</td>
<td>Trestbps</td>
<td>0.85±0.04</td>
<td>0.86±0.02</td>
</tr>
<tr>
<td></td>
<td>Chol.</td>
<td>0.85±0.04</td>
<td>0.85±0.05</td>
</tr>
<tr>
<td></td>
<td>T,C</td>
<td>0.85±0.04</td>
<td>0.86±0.06</td>
</tr>
<tr>
<td>Adult</td>
<td>Cap. Gain</td>
<td>0.84</td>
<td>0.84</td>
</tr>
<tr>
<td></td>
<td>Hours</td>
<td>0.84</td>
<td>0.84</td>
</tr>
</tbody>
</table>

Monotonicity is a *great* inductive bias for learning

COMET: Counterexample-Guided Monotonicity Enforced Training

Table 4: Monotonicity is an effective inductive bias. COMET outperforms Min-Max networks on all datasets. COMET outperforms DLN in regression datasets and achieves similar results in classification datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Features</th>
<th>Min-Max</th>
<th>DLN</th>
<th>COMET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto-</td>
<td>Weight</td>
<td>9.91±1.20</td>
<td>16.77±2.57</td>
<td>8.92±2.93</td>
</tr>
<tr>
<td>MPG</td>
<td>Displ.</td>
<td>11.78±2.20</td>
<td>16.67±2.25</td>
<td>9.11±2.25</td>
</tr>
<tr>
<td></td>
<td>W,D</td>
<td>11.60±0.54</td>
<td>16.56±2.27</td>
<td>8.89±2.29</td>
</tr>
<tr>
<td></td>
<td>W,D,HP</td>
<td>10.14±1.54</td>
<td>13.34±2.42</td>
<td>8.81±1.81</td>
</tr>
<tr>
<td>Boston</td>
<td>Rooms</td>
<td>30.88±13.78</td>
<td>15.93±1.40</td>
<td>11.54±2.55</td>
</tr>
<tr>
<td></td>
<td>Crime</td>
<td>25.89±2.47</td>
<td>12.06±1.44</td>
<td>11.07±2.99</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Features</th>
<th>Min-Max</th>
<th>DLN</th>
<th>COMET</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Trestbps</td>
<td>0.75±0.04</td>
<td>0.85±0.02</td>
<td>0.86±0.03</td>
</tr>
<tr>
<td>Heart</td>
<td>Chol.</td>
<td>0.75±0.04</td>
<td>0.85±0.04</td>
<td>0.87±0.03</td>
</tr>
<tr>
<td></td>
<td>T,C</td>
<td>0.75±0.04</td>
<td>0.86±0.02</td>
<td>0.86±0.03</td>
</tr>
<tr>
<td>Adult</td>
<td>Cap. Gain</td>
<td>0.77</td>
<td>0.84</td>
<td>0.84</td>
</tr>
<tr>
<td></td>
<td>Hours</td>
<td>0.73</td>
<td>0.85</td>
<td>0.84</td>
</tr>
</tbody>
</table>

COMET = Provable Guarantees + SotA Results

The AI Dilemma

Pure (Logic) Reasoning

- Knowledge is (hidden) everywhere in ML
- A little bit of reasoning goes a long way!

Deep learning with structured output constraints
Learning monotonic neural networks
Outline

1. The paradox of learning to reason from data

2. Tractable deep generative models

3. Learning with symbolic knowledge
The AI Dilemma

Pure Reasoning  Pure Learning

Integrate reasoning into modern deep learning algorithms
Thanks

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

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