UCLA Computer Science

STAR AI RESEARCH LAB UCLA

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

Guy Van den Broeck

UCI AI/ML Seminar Series - May 15 2023

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

Outline

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

   ~~deep learning~~

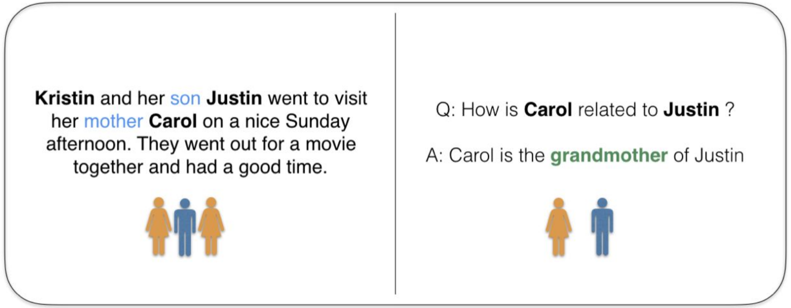
2. Architectures for learning and reasoning

   logical reasoning + deep learning

   a. Constrained language generation
   b. Constrained structured prediction

Can Language Models Perform Logical Reasoning?

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



Reasoning Example from the CLUTRR dataset

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

Language Models:
input → ? → Carol is the grandmother of Justin.

Logical Reasoning:
input → Justin in Kristin’s son; Carol is Kristin’s mother; → Carol is Justin’s mother’s mother; if X is Y’s mother’s mother then X is Y’s grandmother → Carol is the grandmother of Justin.

SimpleLogic

Generate textual train and test examples of the form:

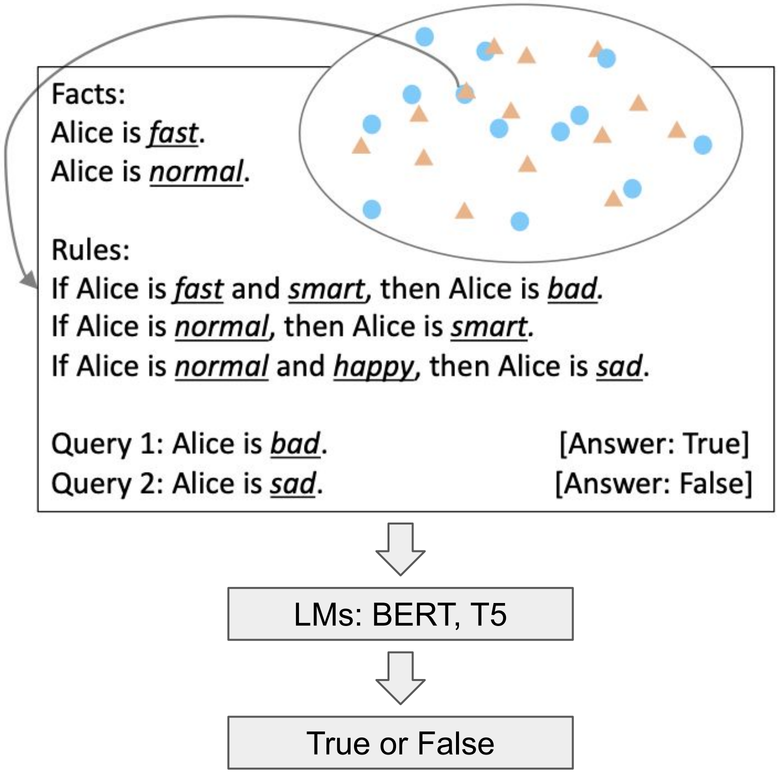
Rules: If witty, then diplomatic. If careless and condemned and attractive, then blushing. If dishonest and inquisitive and average, then shy. If average, then stormy. If popular, then blushing. If talented, then hurt. If popular and attractive, then thoughtless. If blushing and shy and stormy, then 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 ?

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

Problem Setting: SimpleLogic

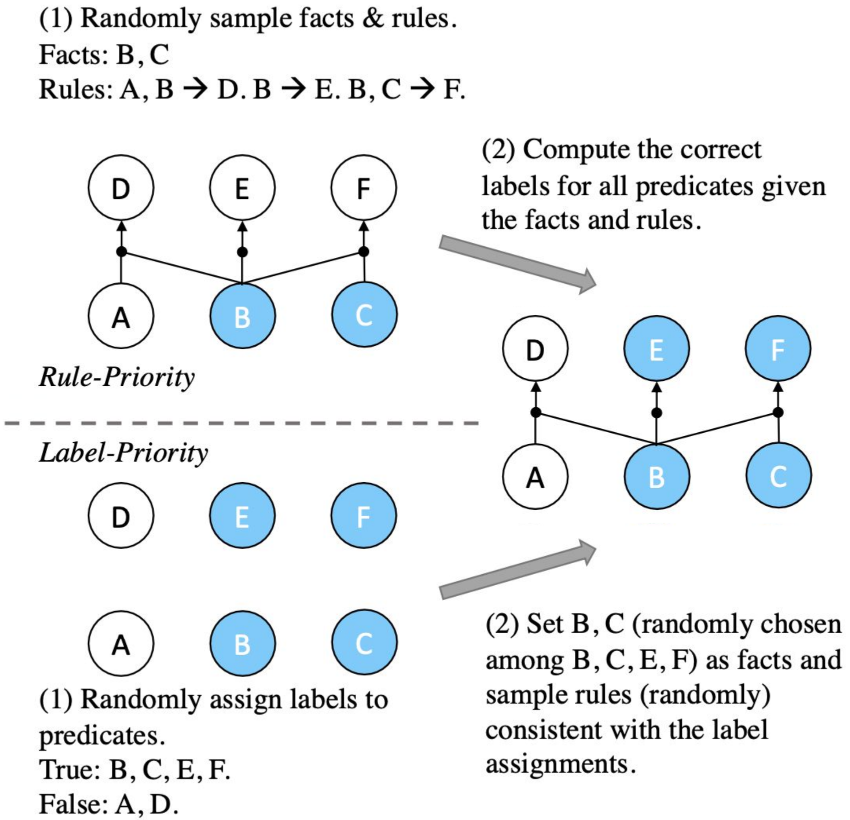
The easiest of reasoning problems:

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



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

Training a transformer on SimpleLogic



Test accuracy for different reasoning depths

Test	0	1	2	3	4	5	6
RP	99.9	99.8	99.7	99.3	98.3	97.5	95.5

Test	0	1	2	3	4	5	6
LP	100.0	100.0	99.9	99.9	99.7	99.7	99.0

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 the transformer 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 transformer 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, the transformer has learned the ground-truth reasoning function!

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

The Paradox of Learning to Reason from Data

Train	Test	0	1	2	3	4	5	6
RP	RP	99.9	99.8	99.7	99.3	98.3	97.5	95.5
	LP	99.8	99.8	99.3	96.0	90.4	75.0	57.3
LP	RP	97.3	66.9	53.0	54.2	59.5	65.6	69.2
	LP	100.0	100.0	99.9	99.9	99.7	99.7	99.0

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

1. If the transformer **has learned** to reason, it should not exhibit such generalization failure.
2. If the transformer **has not learned** to reason, it is baffling how it achieves near-perfect in-distribution test accuracy.

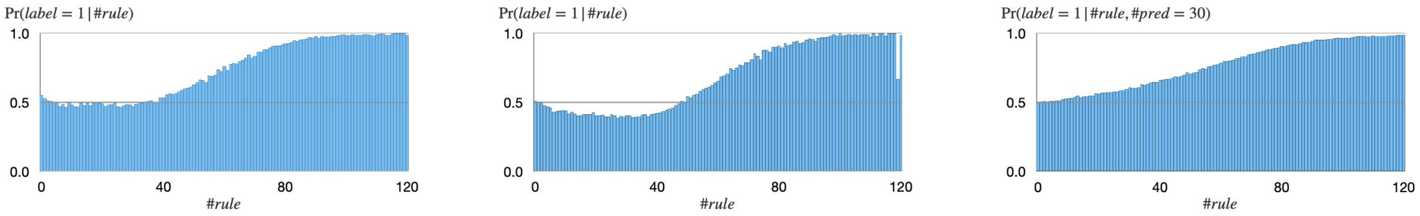
Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang and Guy Van den Broeck. On the Paradox of Learning to Reason from Data, 2022

Why? Statistical Features

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

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

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;

Model leverages statistical features to make predictions

RP_b downsamples from RP such that Pr(label = True | rule# = x) = 0.5 for all x

Train	Test	0	1	2	3	4	5	6
	RP	99.9	99.8	99.7	99.3	98.3	97.5	95.5
RP	RP_b	99.0	99.3	98.5	97.5	96.7	93.5	88.3

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.

FAQ: Do bigger transformers solve this problem? No, already 99% accurate…

FAQ: Will reasoning emerge? Perhaps on 99% of human behavior…

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

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

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

Controlled generation is still 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.

ChatGPT

more reasoning!

Generate image

Stable Diffusion

What do we have?

Prefix: "The weather is"

Constraint α: text contains "winter"

Model only does $p(\text{next-token}|\text{prefix}) =$

cold	0.05
warm	0.10

intractable

Train some $q(.\,|\alpha)$ for a specific task distribution $\alpha \sim p_{\text{task}}$

(amortized inference, encoder, masked model, seq2seq, prompt tuning,...)

Train $q(\text{next-token}|\text{prefix}, \alpha)$

What do we need?

Prefix: "The weather is"

Constraint α: text contains "winter"

Generate from $p(\text{next-token} \mid \text{prefix}, \alpha) =$

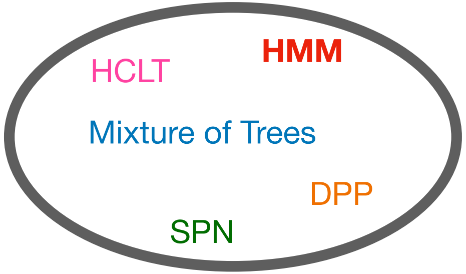
cold	0.50
warm	0.01

$$\propto \sum_{\text{text}} p(\text{next-token}, \text{text}, \text{prefix}, \alpha)$$

Marginalization!

Tractable Probabilistic Models

Probabilistic (Generating) Circuits



Tractable Probabilistic Models (TPMs) model joint probability distributions (just like auto-regressive LMs) and allow efficient computation of various probabilistic queries.

e.g., efficient marginalization:

$$p_{TPM}(\text{3rd token = pan, 5th token = vegetable})$$

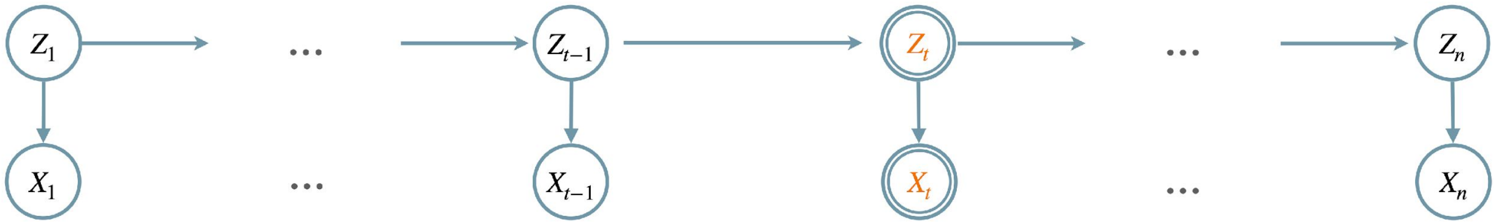
in particular …

$$\sum_{\text{sentence}} p_{TPM}(\textbf{sentence}, \text{next-token = "warm", prefix = "The weather is"}, \alpha)$$

➡ Efficient conditioning given lexical constraints : $p_{TPM}(\text{next-token} \mid \text{prefix}, \alpha)$

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

Step 1: Distill an HMM p_{hmm} that approximates p_{gpt}



1. An HMM with 4096 hidden states and 50k emission tokens
2. Train the HMM on data sampled from GPT2-large (domain-adapted, either via prompting or fine-tuning), effectively minimizing KL(p_{gpt} // p_{HMM})
3. Leverages the latent variable distillation technique for training probabilistic circuits at scale [ICLR 23]. (Cluster embeddings of examples to estimate latent Z_i)

Anji Liu, Honghua Zhang and Guy Van den Broeck. Scaling Up Probabilistic Circuits by Latent Variable Distillation, 2023.

Computing $p_{hmm}(\alpha \mid x_{1:t+1})$

For α in conjunctive normal form (CNF):

$$(w_{1,1} \lor \dots \lor w_{1,d1}) \land \dots \land (w_{m,1} \lor \dots \lor w_{m,dm})$$

where each w_{ij} is a keyword (i.e. a **string of tokens**), representing the constraint that w_{ij} appears in the generated text.

e.g., α = ("swims" ∨ "like swimming") ∧ ("lake" ∨ "pool")

Efficient algorithm:
For m clauses and sequence length n, time-complexity for generation is $O(2^{|m|}n)$.

Trick: dynamic programming with clever preprocessing and local belief updates

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

CommonGen: a Challenging Benchmark

Given 3-5 concepts (keywords), our goal is to generate a sentence using all keywords, which can appear in any order and any form of inflections. e.g.,

Input: snow drive car

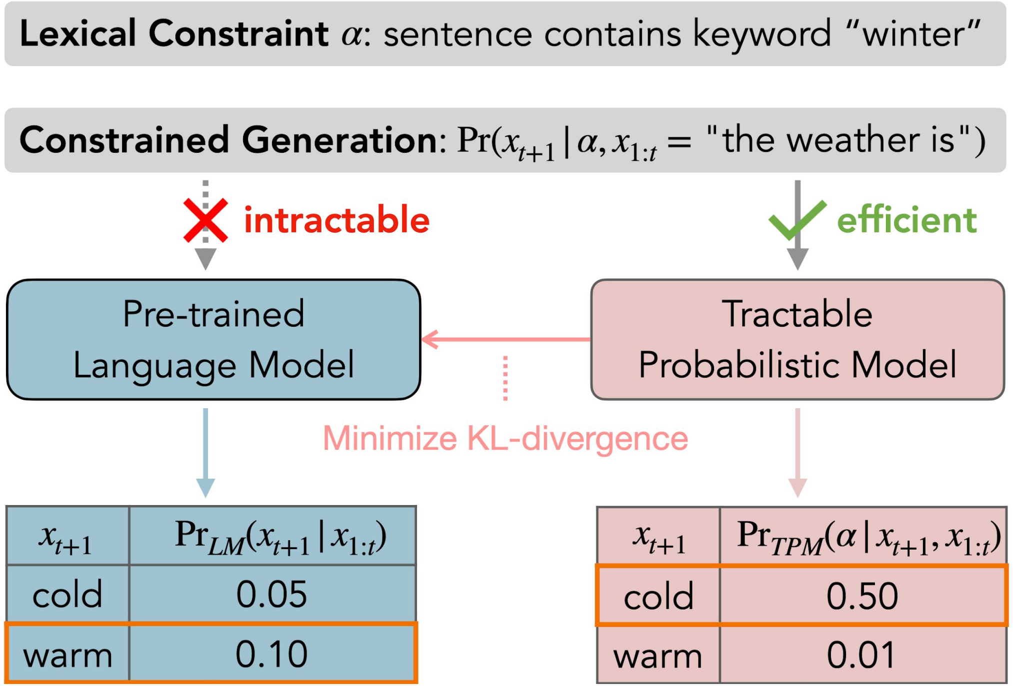
Reference 1: A car drives down a snow covered road.

Reference 2: Two cars drove through the snow.

$$(w_{1,1} \lor \ldots \lor w_{1,d1}) \land \ldots \land (w_{m,1} \lor \ldots \lor w_{m,dm})$$

Each clause represents the inflections for one keyword.

GeLaTo Overview

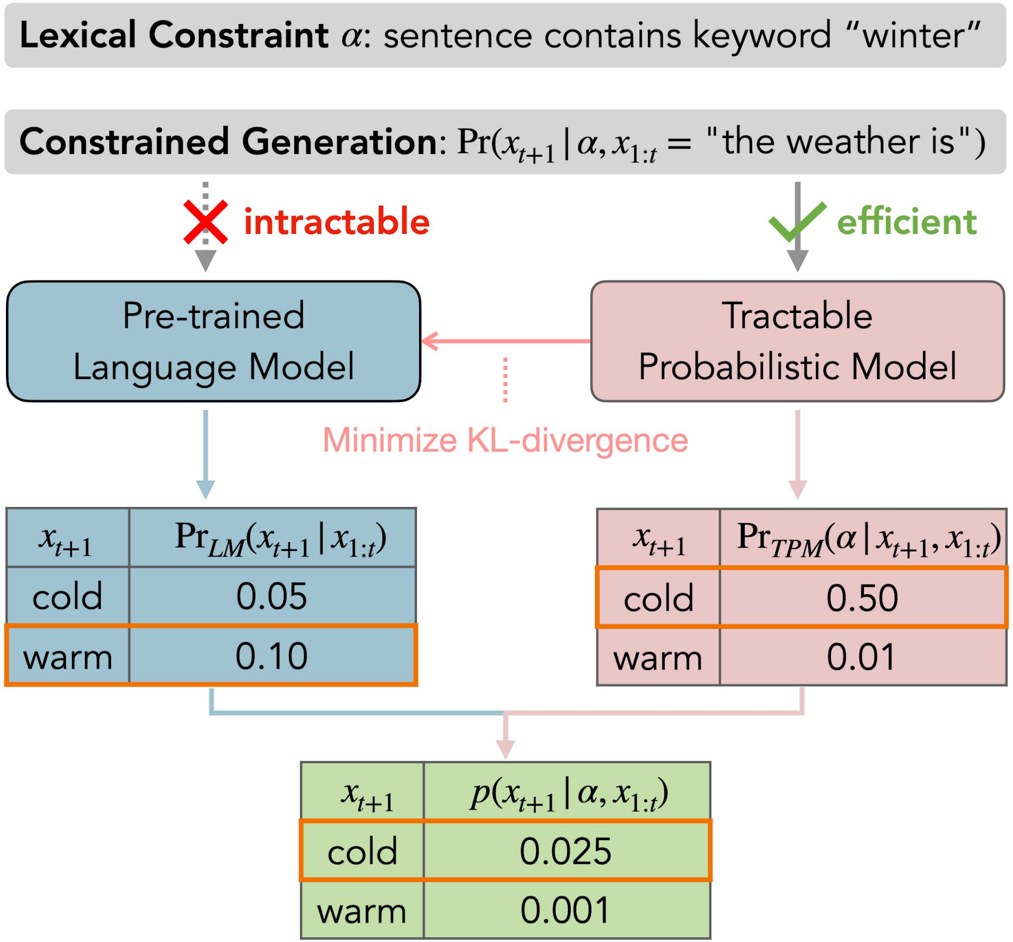


x_{t+1}	$\Pr_{LM}(x_{t+1} \mid x_{1:t})$
cold	0.05
warm	0.10

x_{t+1}	$\Pr_{TPM}(\alpha \mid x_{t+1}, x_{1:t})$
cold	0.50
warm	0.01

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

GeLaTo Overview



x_{t+1}	$\Pr_{LM}(x_{t+1} \mid x_{1:t})$
cold	0.05
warm	0.10

x_{t+1}	$\Pr_{TPM}(\alpha \mid x_{t+1}, x_{1:t})$
cold	0.50
warm	0.01

x_{t+1}	$p(x_{t+1} \mid \alpha, x_{1:t})$
cold	0.025
warm	0.001

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

Step 2: Control p_{gpt} via p_{hmm}

Unsupervised

Language model is not
fine-tuned/prompted to satisfy constraints

By Bayes rule:

$$p_{gpt}(x_{t+1} \mid x_{1:t}, \alpha) \propto p_{gpt}(\alpha \mid x_{1:t+1}) \cdot p_{gpt}(x_{t+1} \mid x_{1:t})$$

Assume $p_{hmm}(\alpha \mid x_{1:t+1}) \approx p_{gpt}(\alpha \mid x_{1:t+1})$, we generate from:

$$p(x_{t+1} \mid x_{1:t}, \alpha) \propto p_{hmm}(\alpha \mid x_{1:t+1}) \cdot p_{gpt}(x_{t+1} \mid x_{1:t})$$

Method	*Generation Quality*								*Constraint Satisfaction*			
	ROUGE-L		BLEU-4		CIDEr		SPICE		Coverage		Success Rate	
Unsupervised	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*
InsNet (Lu et al., 2022a)	-	-	18.7	-	-	-	-	-	**100.0**	-	**100.0**	-
NeuroLogic (Lu et al., 2021)	-	41.9	-	24.7	-	14.4	-	27.5	-	96.7	-	-
A*esque (Lu et al., 2022b)	-	**44.3**	-	28.6	-	15.6	-	29.6	-	97.1	-	-
NADO (Meng et al., 2022)	-	-	26.2	-	-	-	-	-	96.1	-	-	-
GeLaTo	**44.6**	44.1	**29.9**	**29.4**	**16.0**	**15.8**	**31.3**	**31.0**	**100.0**	**100.0**	**100.0**	**100.0**

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

Step 2: Control p_{gpt} via p_{hmm}

Supervised

Language model is fine-tuned to perform constrained generation (e.g. seq2seq)

Empirically $p_{HMM}(\alpha \mid x_{1:t+1}) \approx p_{gpt}(\alpha \mid x_{1:t+1})$ does not hold well enough;

we view $p_{HMM}(x_{t+1} \mid x_{1:t}, \alpha)$ and $p_{gpt}(x_{t+1} \mid x_{1:t})$ as classifiers trained for the same task with different biases; thus we generate from their *weighted geometric mean*:

$$p(x_{t+1} \mid x_{1:t}, \alpha) \propto p_{hmm}(x_{t+1} \mid x_{1:t}, \alpha)^{w} \cdot p_{gpt}(x_{t+1} \mid x_{1:t})^{1-w}$$

Method	Generation Quality								Constraint Satisfaction			
	ROUGE-L		BLEU-4		CIDEr		SPICE		Coverage		Success Rate	
Supervised	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*
NeuroLogic (Lu et al., 2021)	-	42.8	-	26.7	-	14.7	-	30.5	-	97.7	-	93.9†
A*esque (Lu et al., 2022b)	-	43.6	-	28.2	-	15.2	-	30.8	-	97.8	-	97.9†
NADO (Meng et al., 2022)	44.4†	-	30.8	-	16.1†	-	**32.0**†	-	97.1	-	88.8†	-
GeLaTo	**46.0**	**45.6**	**34.1**	**32.9**	**16.7**	**16.8**	31.3	**31.9**	**100.0**	**100.0**	**100.0**	**100.0**

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

Advantages of our framework:

1. Constraint α is <u>guaranteed to be satisfied</u>: for any next-token x_{t+1} that would make α unsatisfiable, $p(x_{t+1} \mid x_{1:t}, \alpha) = 0$ for both settings.

2. Training p_{hmm} <u>does not depend on α</u>, which is only imposed at inference (generation) time. Once p_{hmm} is trained, we can impose whatever α.

3. We can impose <u>additional tractable constraints</u>:
   - The keywords are generated following a particular order.
   - (Some) keywords must appear at a particular position.
   - (Some) keywords must not appear in the generated sentence.

Conclusion: control intractable generative model by tractable generative model for (symbolic) reasoning.

Outline

1. The paradox of learning to reason from data

   ~~deep learning~~

2. **Architectures for learning and reasoning**

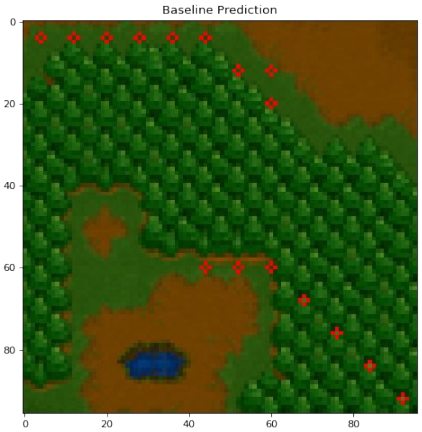
   logical reasoning + deep learning

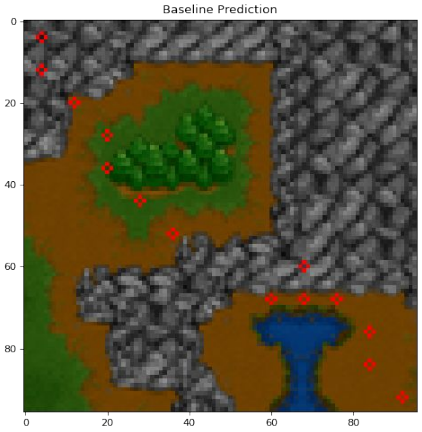
   a. Constrained language generation
   b. Constrained structured prediction

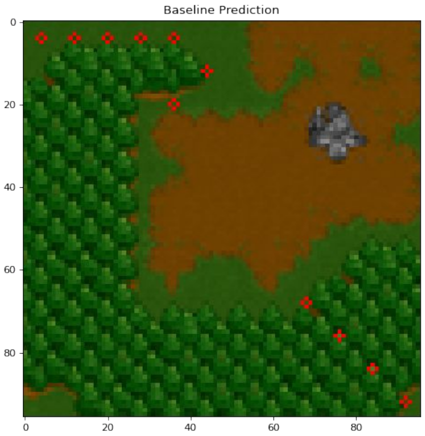
Warcraft Shortest Path

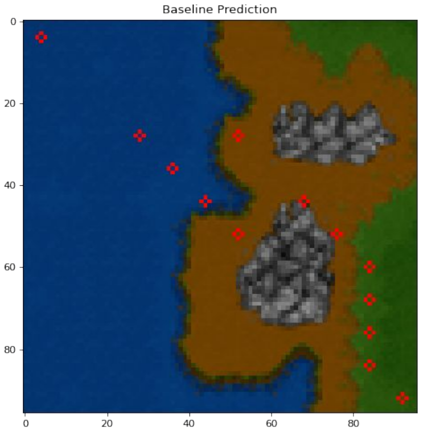
// for a 12×12 *grid,* 2^{144} *states but only* 10^{10} *valid ones!*

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


Baseline Prediction
0
20
40
60
80


Baseline Prediction
0
20
40
60
80


Baseline Prediction
0
20
40
60
80


Baseline Prediction
0
20
40
60
80

Architecture	Exact Match	Hamming Score	Consistency
ResNet-18+FIL	55.0	**97.7**	56.9

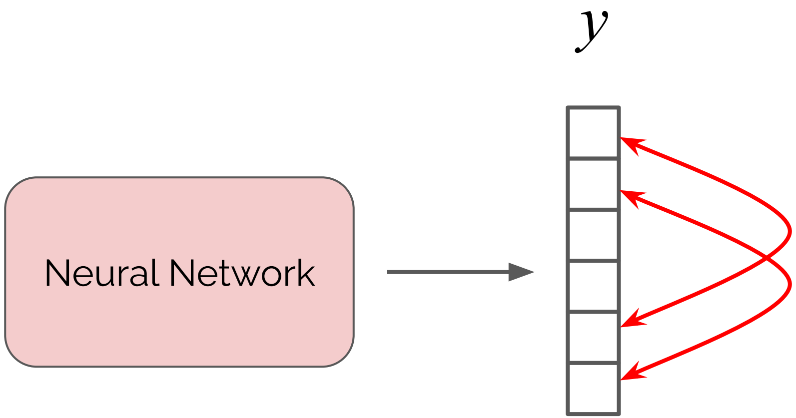
Is prediction the shortest path?
This is the real task!

Are individual edge predictions correct?

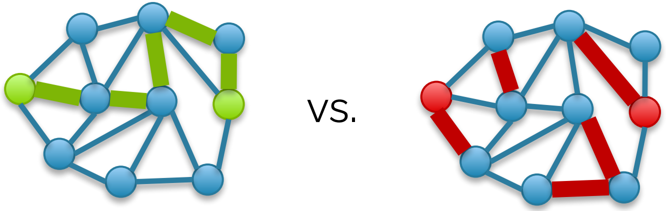
Is output a path?

Kareem Ahmed, Eric Wang, Kai-Wei Chang and Guy Van den Broeck. Neuro-Symbolic Entropy Regularization, 2021.

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

Kareem Ahmed, Tao Li, Thy Ton, Quan Guo, Kai-Wei Chang, Parisa Kordjamshidi, Vivek Srikumar, Guy Van den Broeck and Sameer Singh. PYLON: A PyTorch Framework for Learning with Constraints

pylon

PyTorch Code

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

1 Specify knowledge as a predicate

```
def check(y):
    ...
    return isValid
```

Kareem Ahmed, Tao Li, Thy Ton, Quan Guo, Kai-Wei Chang, Parisa Kordjamshidi, Vivek Srikumar, Guy Van den Broeck and Sameer Singh. PYLON: A PyTorch Framework for Learning with Constraints

pylon

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)
```

Kareem Ahmed, Tao Li, Thy Ton, Quan Guo, Kai-Wei Chang, Parisa Kordjamshidi, Vivek Srikumar, Guy Van den Broeck and Sameer Singh. PYLON: A PyTorch Framework for Learning with Constraints

pylon

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

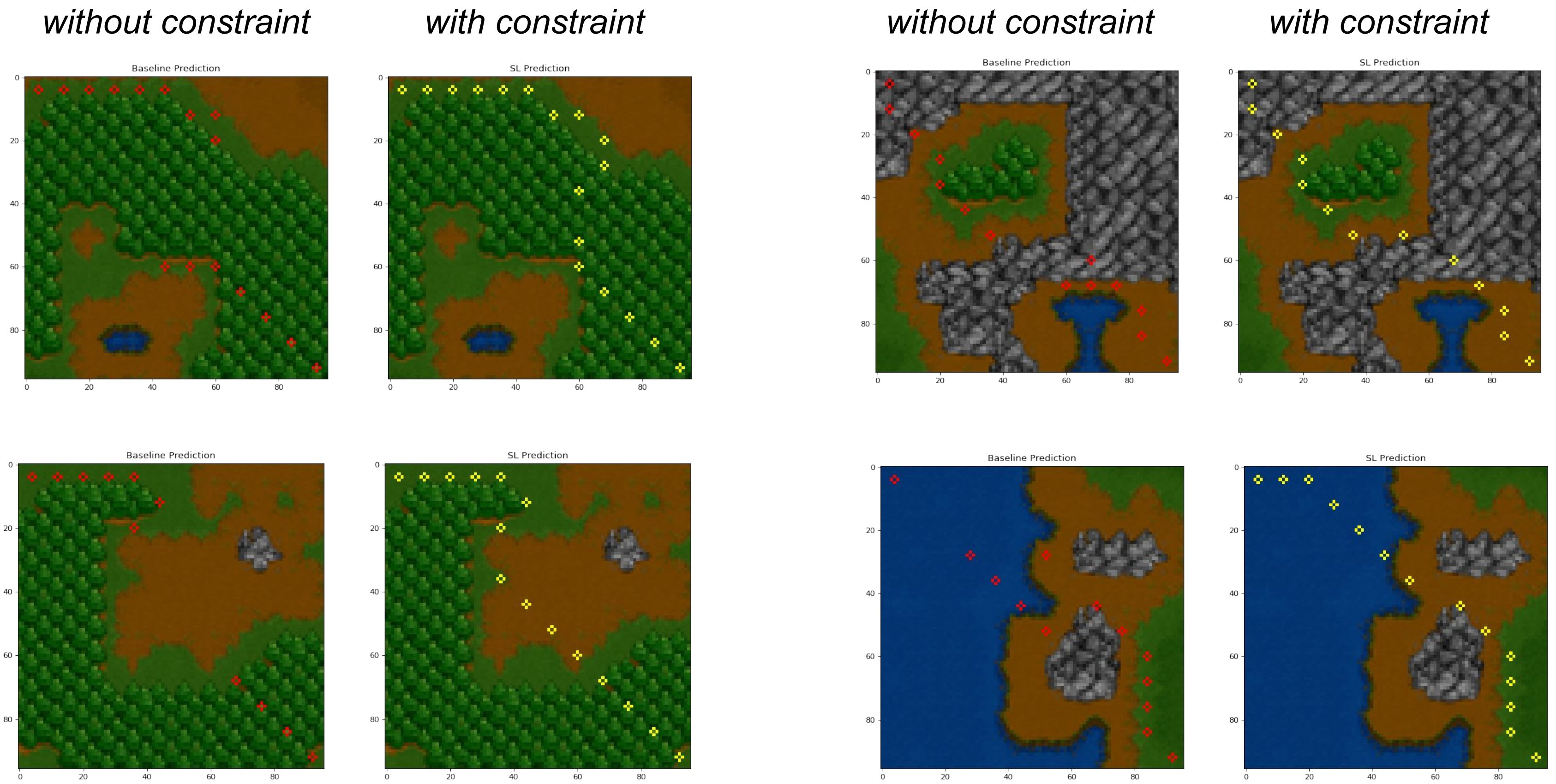
```
def check(y):
    ...
    return isValid
```

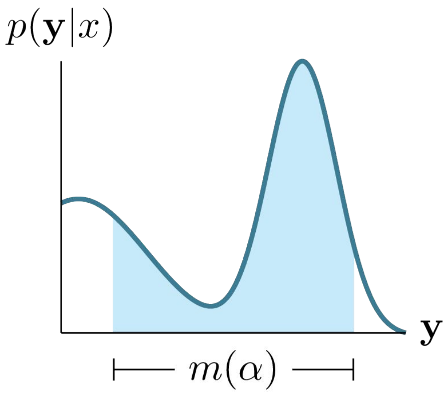
2 Add as loss to training

```
loss += constraint_loss(check)
```

3 pylon derives the gradients
(solves a combinatorial problem)

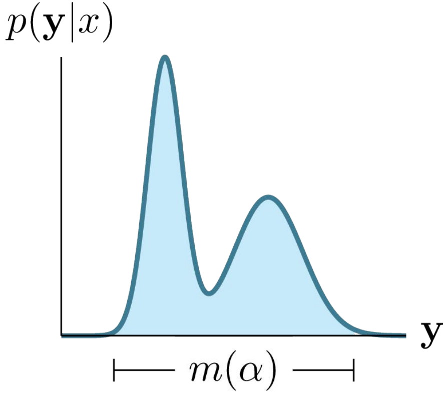
Kareem Ahmed, Tao Li, Thy Ton, Quan Guo, Kai-Wei Chang, Parisa Kordjamshidi, Vivek Srikumar, Guy Van den Broeck and Sameer Singh. PYLON: A PyTorch Framework for Learning with Constraints


without constraint
with constraint
without constraint
with constraint
Baseline Prediction
SL Prediction
Baseline Prediction
SL Prediction
Baseline Prediction
SL Prediction
Baseline Prediction
SL Prediction



a) A network uncertain over both valid & invalid predictions

Semantic Loss



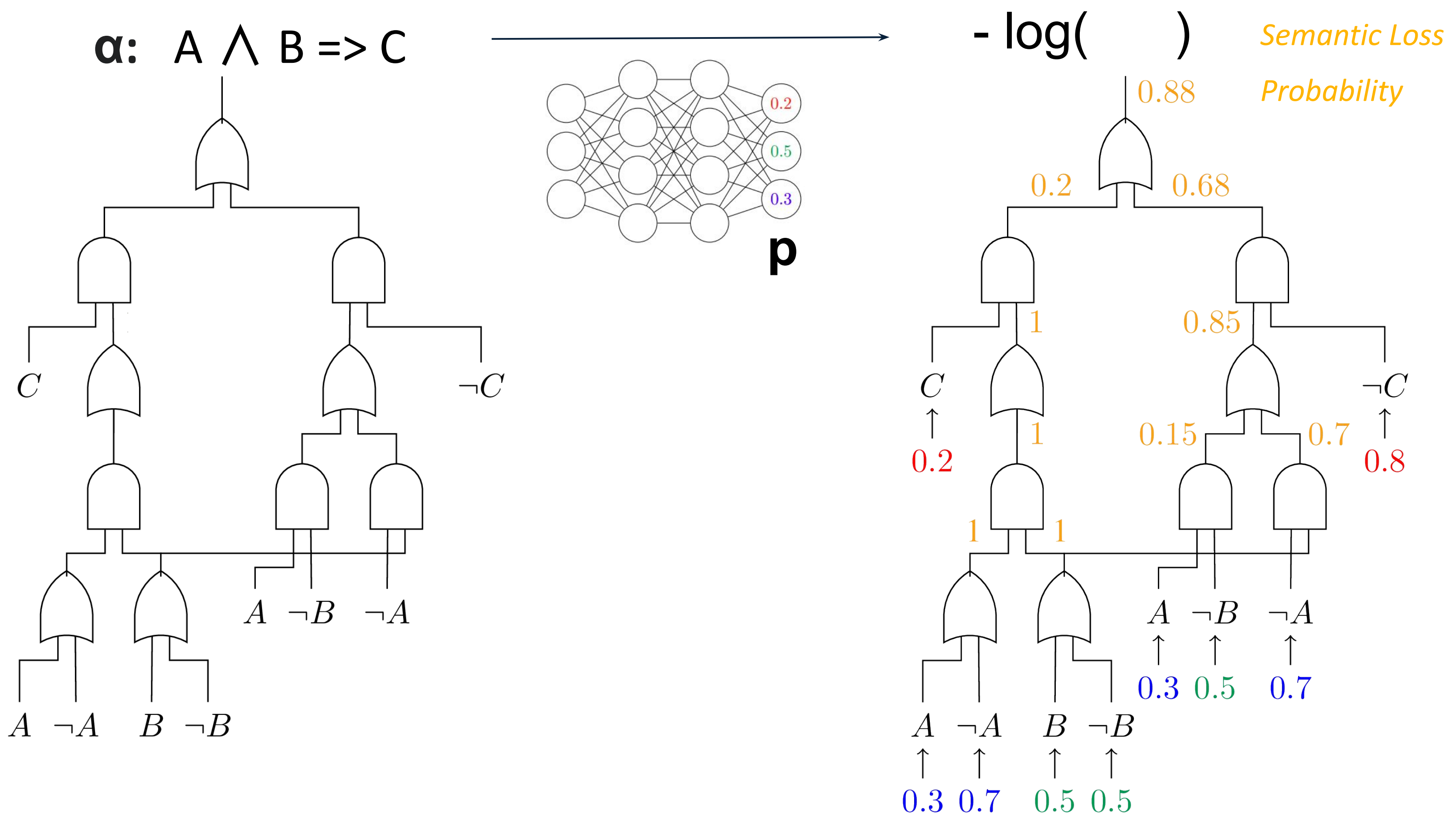
c) A network allocating most of its mass to models of constraint

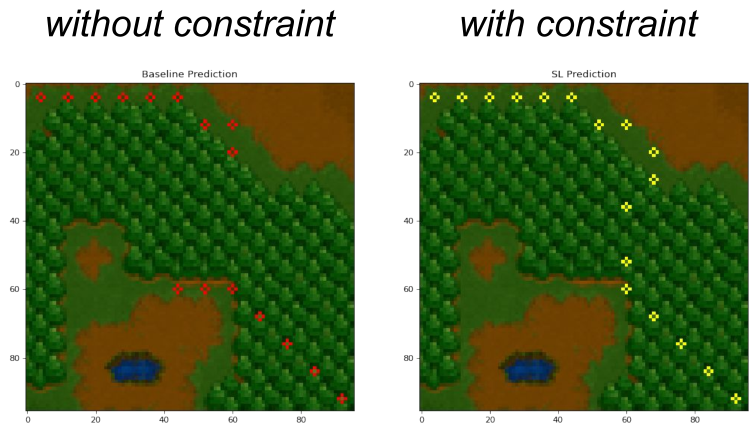
$$\mathrm{L}^s(\alpha, \mathsf{p}) \propto -\log \sum_{\mathbf{x} \models \alpha} \prod_{i:\mathbf{x} \models X_i} \mathsf{p}_i \prod_{i:\mathbf{x} \models \neg X_i} (1 - \mathsf{p}_i)$$

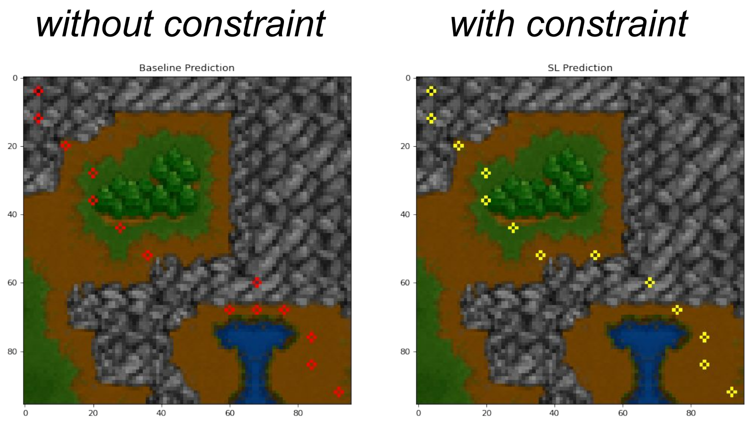
Probability of satisfying constraint α after sampling from neural net output layer **p**

In general: #P-hard ☹

Do this probabilistic-logical reasoning during learning in a computation graph


α: A ∧ B => C
- log( )
Semantic Loss
Probability
p
0.2
0.5
0.3
0.88
0.2
0.68
1
0.85
1
0.15
0.7
1
1
C
¬C
0.2
0.8
A ¬B ¬A
0.3 0.5 0.7
A ¬A B ¬B
0.3 0.7 0.5 0.5

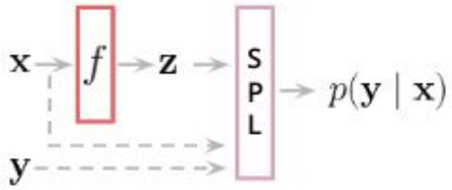


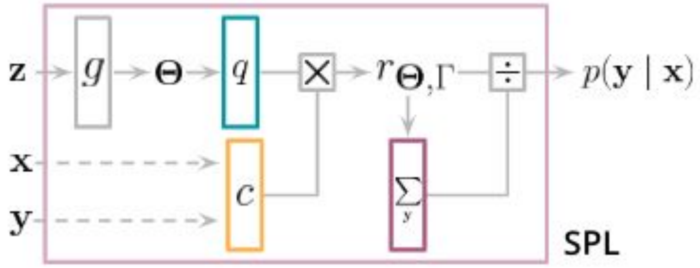


Architecture	Exact Match	Hamming Score	Consistency
ResNet-18+FIL	55.0	**97.7**	56.9
ResNet-18+$\mathcal{L}_{\text{SL}}$	59.4	**97.7**	61.2

Semantic Probabilistic Layers

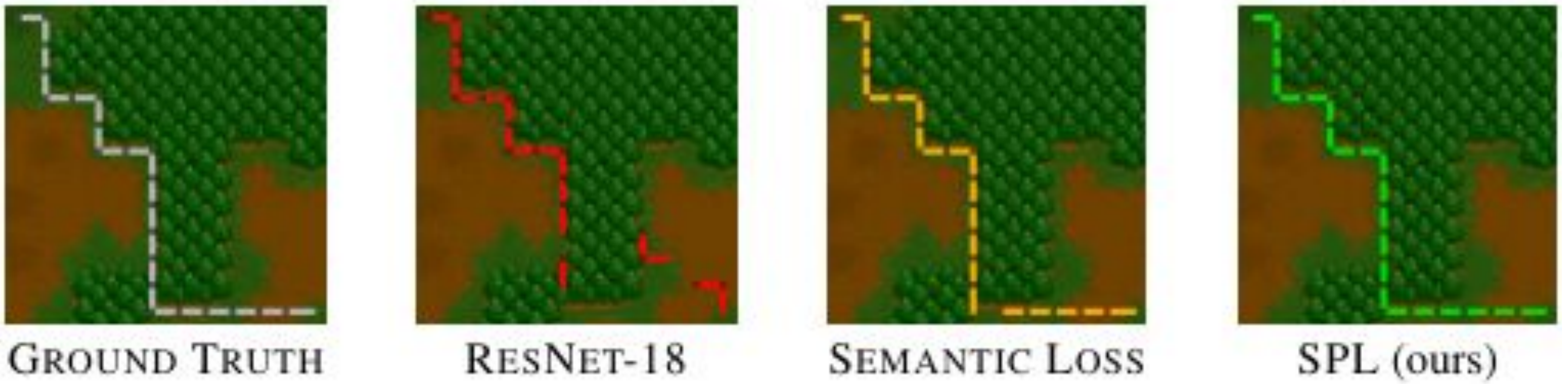
- How to give a 100% guarantee that Boolean constraints will be satisfied?
- Bake the constraint into the neural network as a special layer





- Secret sauce is again tractable circuits – computation graphs for reasoning

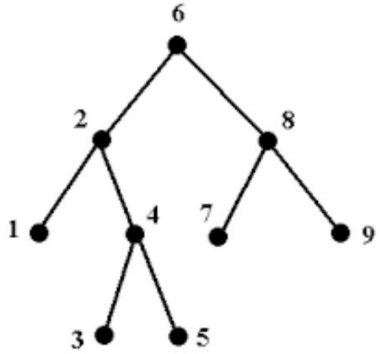
Kareem Ahmed, Stefano Teso, Kai-Wei Chang, Guy Van den Broeck and Antonio Vergari. Semantic Probabilistic Layers for Neuro-Symbolic Learning, 2022.



ARCHITECTURE	EXACT MATCH	HAMMING SCORE	CONSISTENCY
RESNET-18+FIL	55.0	**97.7**	56.9
RESNET-18+$\mathcal{L}_{\mathrm{SL}}$	59.4	**97.7**	61.2
RESNET-18+SPL	75.1	97.6	**100.0**
OVERPARAM. SDD	**78.2**	96.3	**100.0**

Kareem Ahmed, Stefano Teso, Kai-Wei Chang, Guy Van den Broeck and Antonio Vergari. Semantic Probabilistic Layers for Neuro-Symbolic Learning, 2022.

Hierarchical Multi-Label Classification

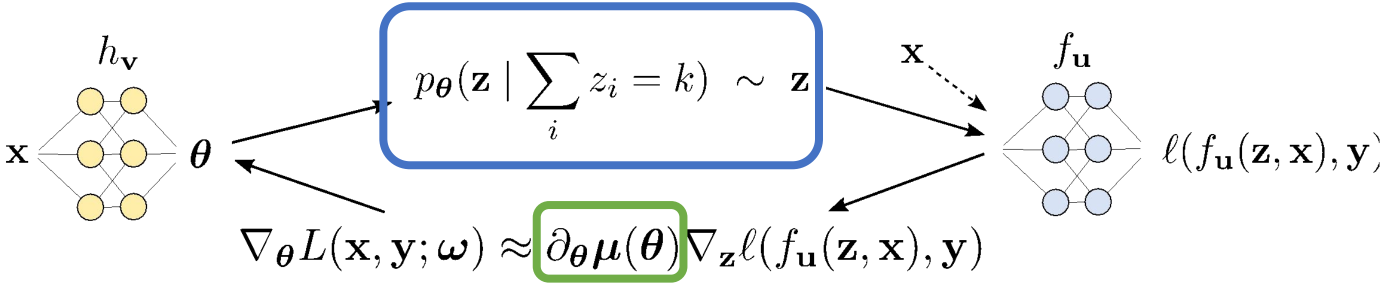


“if the image is classified as a dog, it must also be classified as an animal”

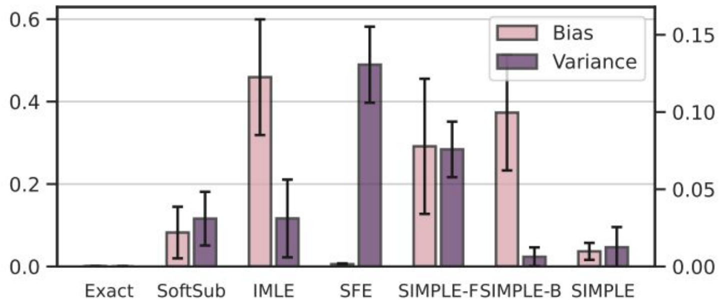
“if the image is classified as an animal, it must be classified as either cat or dog”

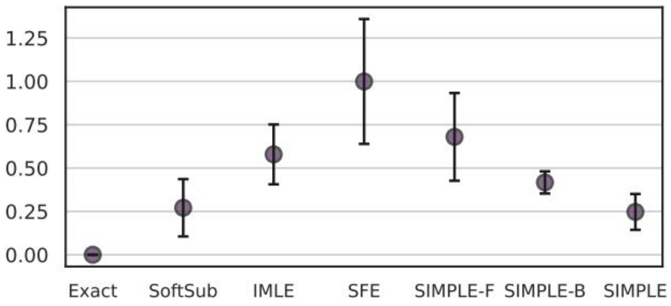
DATASET	EXACT MATCH	
	HMCNN	MLP+SPL
CELLCYCLE	3.05 ± 0.11	$\mathbf{3.79 \pm 0.18}$
DERISI	1.39 ± 0.47	$\mathbf{2.28 \pm 0.23}$
EISEN	5.40 ± 0.15	$\mathbf{6.18 \pm 0.33}$
EXPR	4.20 ± 0.21	$\mathbf{5.54 \pm 0.36}$
GASCH1	3.48 ± 0.96	$\mathbf{4.65 \pm 0.30}$
GASCH2	3.11 ± 0.08	$\mathbf{3.95 \pm 0.28}$
SEQ	5.24 ± 0.27	$\mathbf{7.98 \pm 0.28}$
SPO	$\mathbf{1.97 \pm 0.06}$	1.92 ± 0.11
DIATOMS	48.21 ± 0.57	$\mathbf{58.71 \pm 0.68}$
ENRON	5.97 ± 0.56	$\mathbf{8.18 \pm 0.68}$
IMCLEF07A	79.75 ± 0.38	$\mathbf{86.08 \pm 0.45}$
IMCLEF07D	76.47 ± 0.35	$\mathbf{81.06 \pm 0.68}$

SIMPLE: Gradient Estimator for *k*-Subset Sampling



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





and SotA Learning to Explain (L2X) and sparse discrete VAE results.

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

Secret Sauce: Probabilistic Circuits

Tutorial (3h)

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

Overview Paper (80p)

Probabilistic Circuits:
A Unifying Framework for Tractable Probabilistic Models*

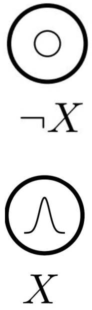
YooJung Choi
Antonio Vergari
Guy Van den Broeck
Computer Science Department
University of California
Los Angeles, CA, USA

Contents

http://starai.cs.ucla.edu/papers/ProbCirc20.pdf

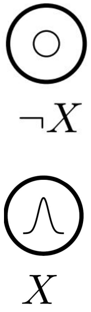
Probabilistic circuits

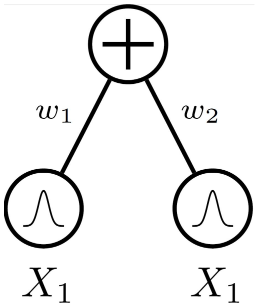
computational graphs that recursively define distributions



Probabilistic circuits

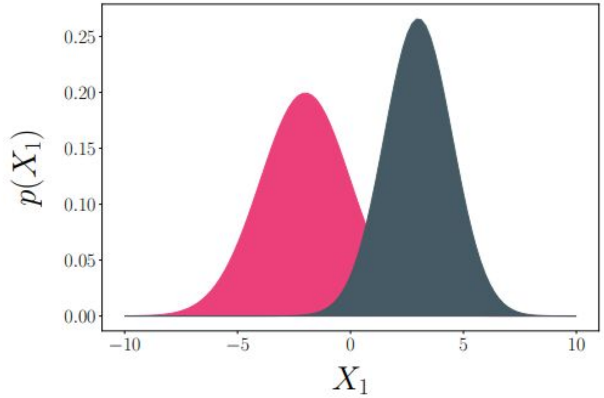
computational graphs that recursively define distributions





$$p(X_1) = w_1 p_1(X_1) + w_2 p_2(X_1)$$

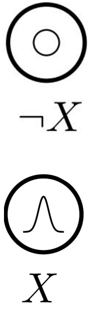
$\Rightarrow$
mixtures

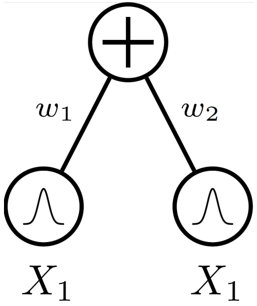


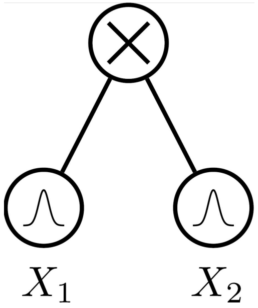
$$p(X) = p(Z = \boxed{1}) \cdot p_1(X|Z = \boxed{1})$$
$$+\, p(Z = \boxed{2}) \cdot p_2(X|Z = \boxed{2})$$

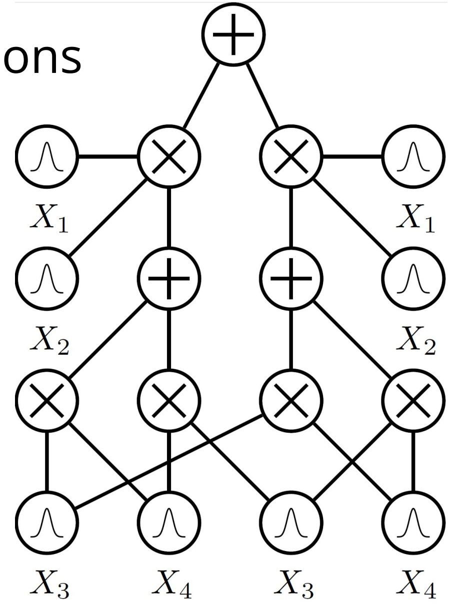
Probabilistic circuits

computational graphs that recursively define distributions









$$p(X_1) = w_1 p_1(X_1) + w_2 p_2(X_1)$$

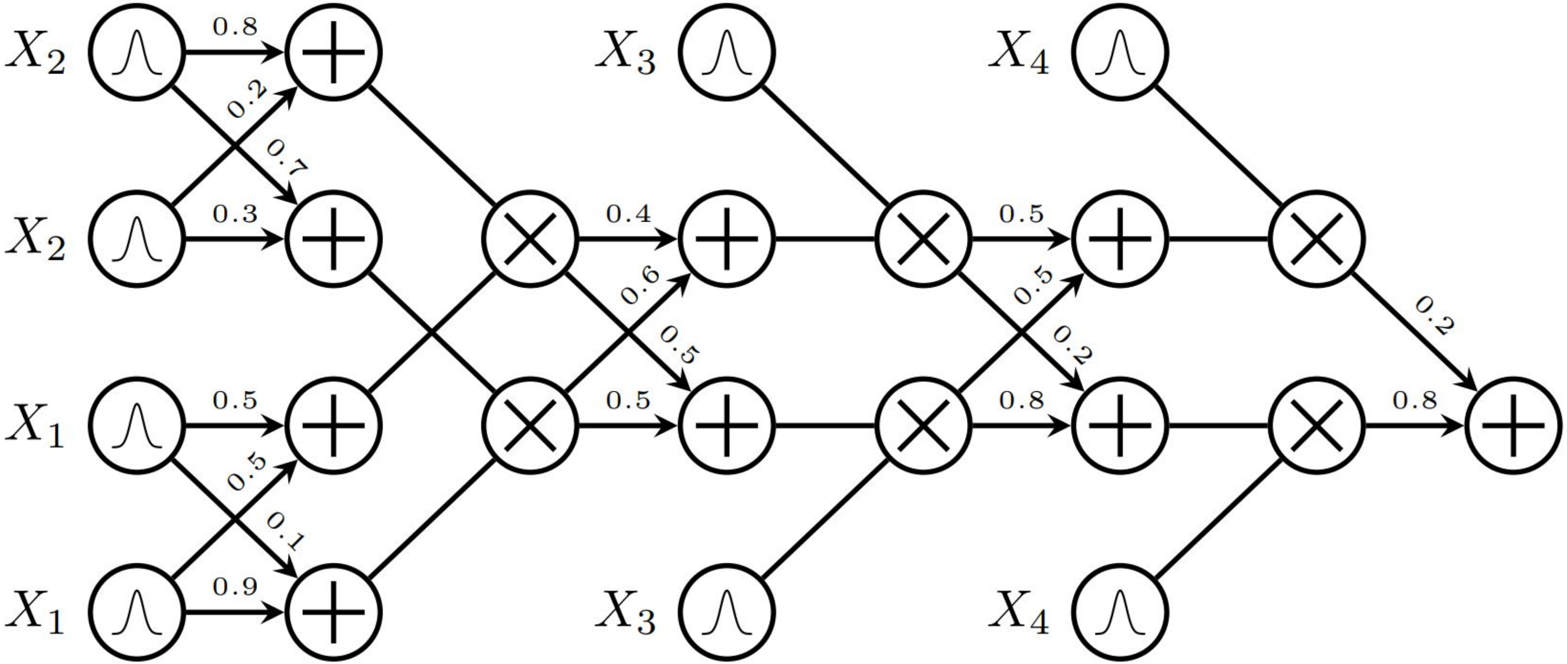
$\Rightarrow$ *mixtures*

$$p(X_1, X_2) = p(X_1) \cdot p(X_2)$$

$\Rightarrow$ *factorizations*

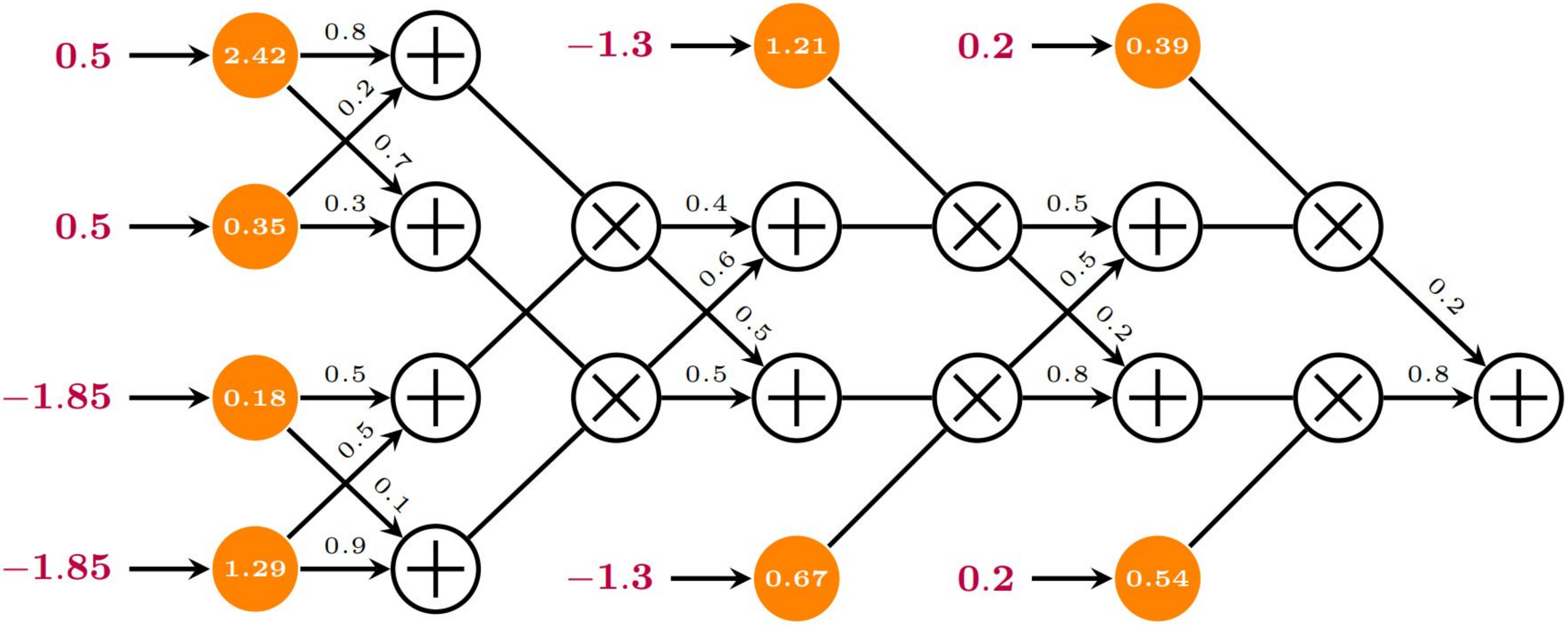
Likelihood

$$p(X_1 = \mathbf{-1.85}, X_2 = \mathbf{0.5}, X_3 = \mathbf{-1.3}, X_4 = \mathbf{0.2})$$



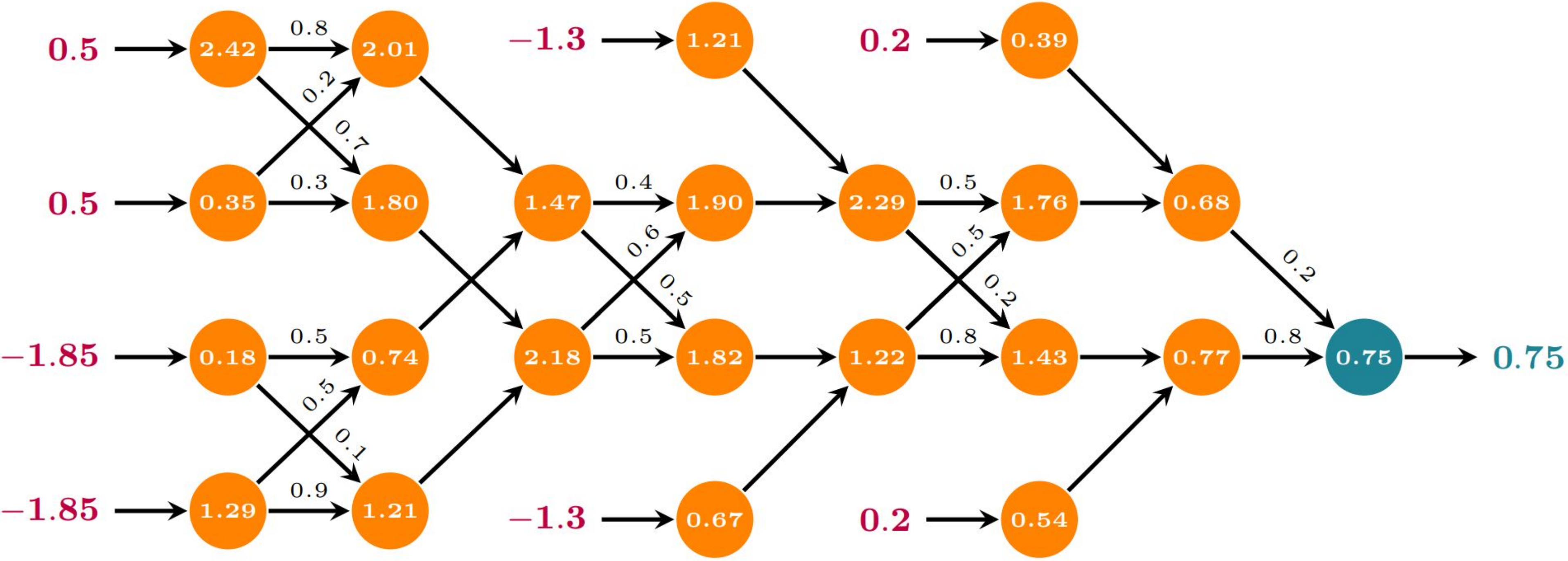
Likelihood

$$p(X_1 = \mathbf{-1.85}, X_2 = \mathbf{0.5}, X_3 = \mathbf{-1.3}, X_4 = \mathbf{0.2})$$



Likelihood

$$p(X_1 = \mathbf{-1.85}, X_2 = \mathbf{0.5}, X_3 = \mathbf{-1.3}, X_4 = \mathbf{0.2})$$



Smoothness + **decomposability** = **tractable MAR**

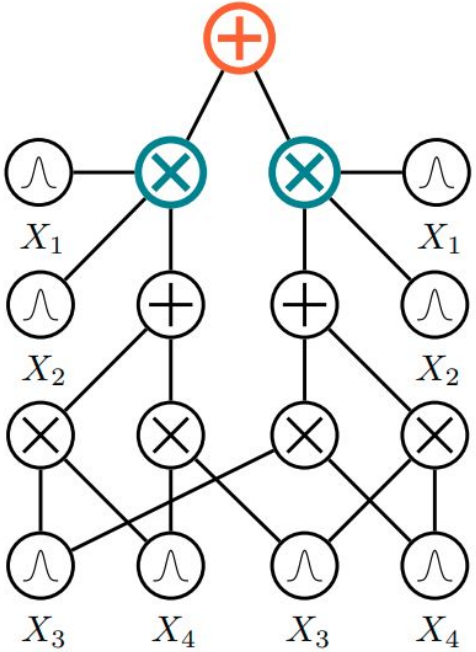
Smoothness + decomposability = tractable MAR

If $p(\mathbf{x}) = \sum_i w_i p_i(\mathbf{x})$, (***smoothness***):

$$\int p(\mathbf{x}) d\mathbf{x} = \int \sum_i w_i p_i(\mathbf{x}) d\mathbf{x} =$$

$$= \sum_i w_i \int p_i(\mathbf{x}) d\mathbf{x}$$

$\Rightarrow$ *integrals are "pushed down" to children*



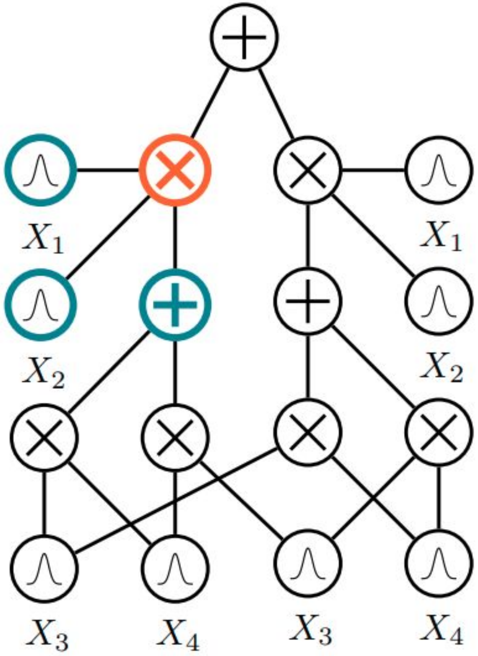
[Darwiche & Marquis JAIR 2001, Poon & Domingos UAI11]

Smoothness + decomposability = tractable MAR

If $p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{x})p(\mathbf{y})p(\mathbf{z})$, (***decomposability***):

$$\int\int\int p(\mathbf{x}, \mathbf{y}, \mathbf{z}) d\mathbf{x} d\mathbf{y} d\mathbf{z} =$$
$$= \int\int\int p(\mathbf{x})p(\mathbf{y})p(\mathbf{z}) d\mathbf{x} d\mathbf{y} d\mathbf{z} =$$
$$= \int p(\mathbf{x}) d\mathbf{x} \int p(\mathbf{y}) d\mathbf{y} \int p(\mathbf{z}) d\mathbf{z}$$

$\Longrightarrow$ *integrals decompose into easier ones*



Smoothness + decomposability = tractable MAR

Forward pass evaluation for MAR

$\Rightarrow$ *linear in circuit size!*

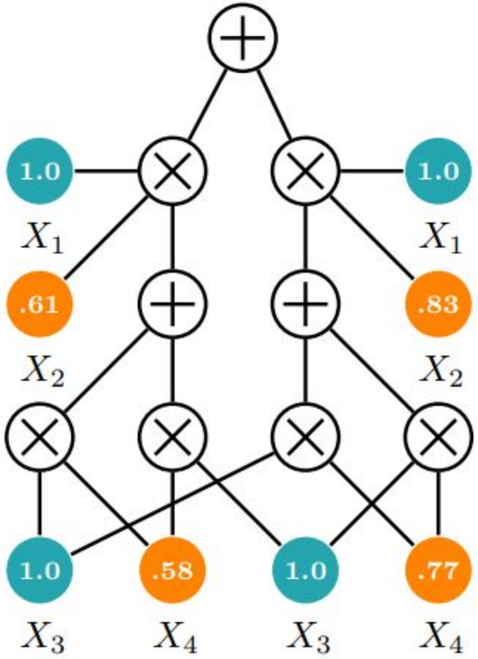
E.g. to compute $p(x_2, x_4)$:

- leafs over X_1 and X_3 output $\boldsymbol{Z}_i = \int p(x_i) dx_i$

  $\Rightarrow$ *for normalized leaf distributions:* 1.0

- leafs over X_2 and X_4 output ***EVI***

- feedforward evaluation (bottom-up)



Smoothness + decomposability = tractable MAR

Forward pass evaluation for MAR

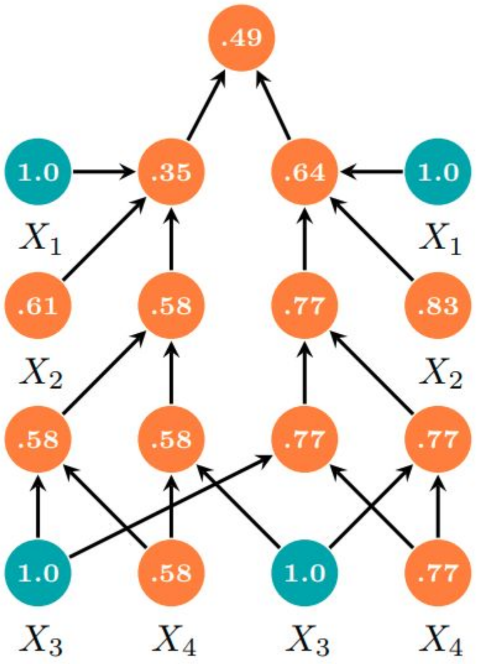
$\Rightarrow$ *linear in circuit size!*

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

- leafs over X_1 and X_3 output $\boldsymbol{Z}_i = \int p(x_i)dx_i$

  $\Rightarrow$ *for normalized leaf distributions:* 1.0

- leafs over X_2 and X_4 output ***EVI***
- feedforward evaluation (bottom-up)



Cute, but these models cannot compete?

bpd	2008-2020
Tabular	😐
MNIST	😱
F-MNIST	😱
EMNIST-L	😱
CIFAR	😱
Imagenet32	😱
Imagenet64	😱

Cute, but these models cannot compete?

bpd	2008-2020	2020-2021
Tabular	😐	😊
MNIST	😱	😱 > 1.67
F-MNIST	😱	😱 > 4.29
EMNIST-L	😱	😱 > 2.73
CIFAR	😱	😱
Imagenet32	😱	😱
Imagenet64	😱	😱

General-purpose architecture

Cute, but these models cannot compete?

	2008-2020	2020-2021	ICLR 22
Tabular	😐	😊	🍰
MNIST	😱	😱 > 1.67	1.20
F-MNIST	😱	😱 > 4.29	3.34
EMNIST-L	😱	😱 > 2.73	1.80
CIFAR	😱	😱	😱 > 5.50
Imagenet32	😱	😱	😱
Imagenet64	😱	😱	😱

General-purpose architecture

Custom GPU kernels

Cute, but these models cannot compete?

	2008-2020	2020-2021	ICLR 22	NeurIPS 22
Tabular	😐	😊	🍰	🍰
MNIST	😱	😱 > 1.67	1.20	1.14
F-MNIST	😱	😱 > 4.29	3.34	3.27
EMNIST-L	😱	😱 > 2.73	1.80	1.58
CIFAR	😱	😱	😱 > 5.50	😱
Imagenet32	😱	😱	😱	😱
Imagenet64	😱	😱	😱	😱

General-purpose architecture

Custom GPU kernels

Pruning without losing likelihood

Cute, but these models cannot compete?

	2008-2020	2020-2021	ICLR 22	NeurIPS 22
Tabular	😐	😊	🍰	🍰
MNIST	😱	😱 > 1.67	1.20	1.14
F-MNIST	😱	😱 > 4.29	3.34	3.27
EMNIST-L	😱	😱 > 2.73	1.80	1.58
CIFAR	😱	😱	😱 > 5.50	😱
Imagenet32	😱	😱	😱	😱
Imagenet64	😱	😱	😱	😱

	Discrete Flow	Hierarchical VAE	PixelVAE
MNIST	1.90	1.27	1.39
F-MNIST	3.47	3.28	3.66
EMNIST-L	1.95	1.84	2.26

Cute, but these models cannot compete?

	2008-2020	2020-2021	ICLR 22	NeurIPS 22	ICLR 23
Tabular	😐	😊	🍰	🍰	🍰
MNIST	😱	😱 > 1.67	1.20	1.14	🍰
F-MNIST	😱	😱 > 4.29	3.34	3.27	🍰
EMNIST-L	😱	😱 > 2.73	1.80	1.58	🍰
CIFAR	😱	😱	😱 > 5.50	😱	4.38
Imagenet32	😱	😱	😱	😱	4.39
Imagenet64	😱	😱	😱	😱	4.12

General-purpose architecture

Custom GPU kernels

Pruning without losing likelihood

Latent Variable Distillation

Cute, but these models cannot compete?

	2008-2020	2020-2021	ICLR 22	NeurIPS 22	ICLR 23	ICML 23
Tabular	😐	😊	🍰	🍰	🍰	🍰
MNIST	😱	😱 > 1.67	1.20	1.14	🍰	🍰
F-MNIST	😱	😱 > 4.29	3.34	3.27	🍰	🍰
EMNIST-L	😱	😱 > 2.73	1.80	1.58	🍰	🍰
CIFAR	😱	😱	😱 > 5.50	😱	4.38	3.87
Imagenet32	😱	😱	😱	😱	4.39	4.06
Imagenet64	😱	😱	😱	😱	4.12	3.80

	Flow	Hierarchical VAE	Diffusion
CIFAR	3.35	3.08	2.65
Imagenet32	4.09	3.96	3.72
Imagenet64	3.81	-	3.40

Secret Sauce: Probabilistic Circuits

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

http://starai.cs.ucla.edu/papers/ProbCirc20.pdf

Outline

1. The paradox of learning to reason from data

   ~~deep learning~~

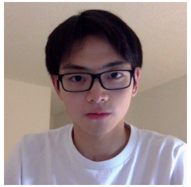
2. Architectures for learning and reasoning

   logical (and probabilistic) reasoning + *deep learning*

   a. Constrained language generation
   b. Constrained structured prediction

Thanks

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



Honghua

Kareem

Zhe

Meihua

Anji

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