Introduction to Logic Reasoning

2022 OxML Summer School

June 29, 2022

The Band

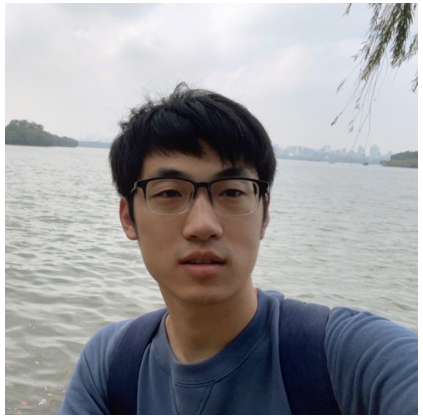
Speaker

Yitao Liang
Assistant Professor, PKU

TAs

Anji Liu
UCLA

Zihao Wang
PKU



Shaofei Cai
PKU

Objectives

To get you started on using logic in ML

A representation / framework

Three libraries

The AI Dilemma

Pure (Logic) Reasoning

Pure Learning

The AI Dilemma

Pure (Logic) Reasoning **Pure Learning**

- 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

The AI Dilemma

Pure (Logic) Reasoning — **Pure Learning**

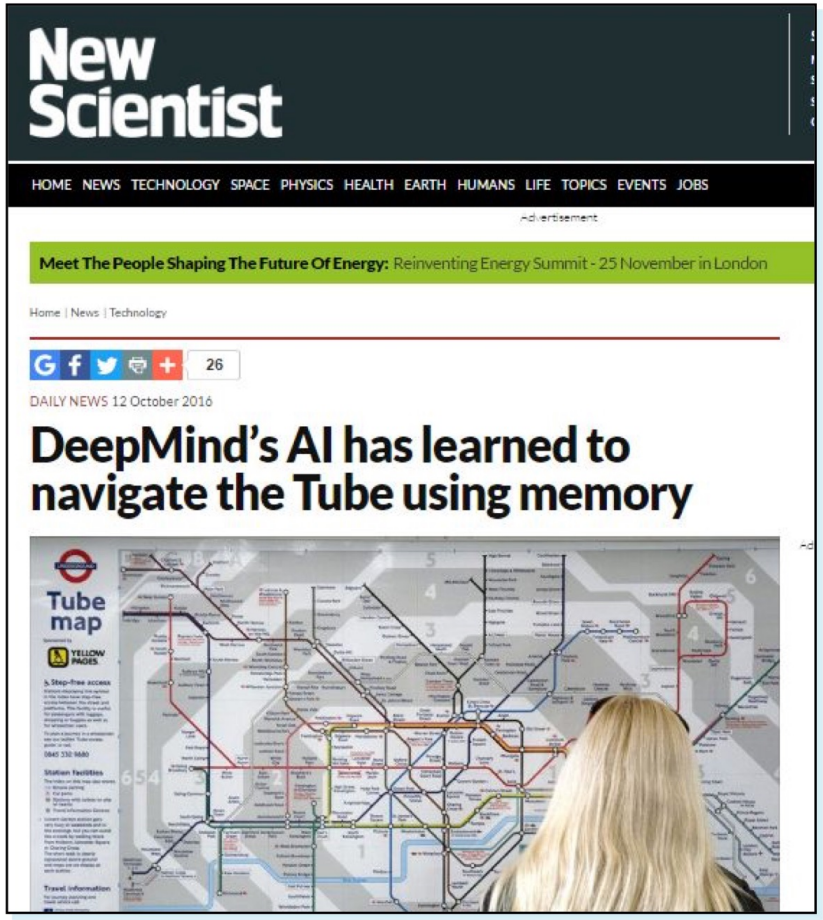
- Slow thinking: deliberative, cognitive, model-based, extrapolation
- Amazing achievements until this day

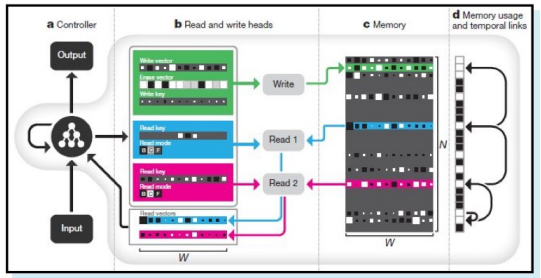
- "*Pure logic is brittle*"
  noise, uncertainty, incomplete knowledge, …

The AI Dilemma

Integrate reasoning into learning

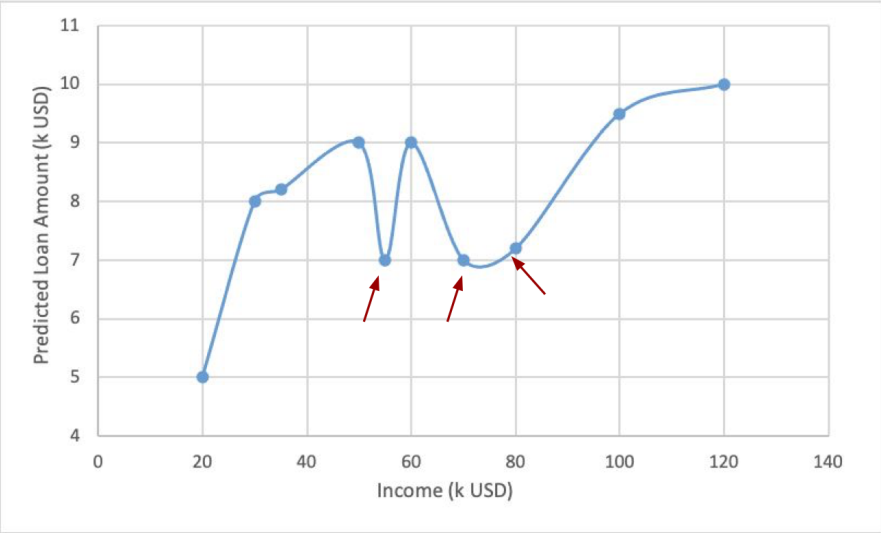
What is the Role of Logic and Reasoning





[Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., et al.. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, *538*(7626), 471-476.]

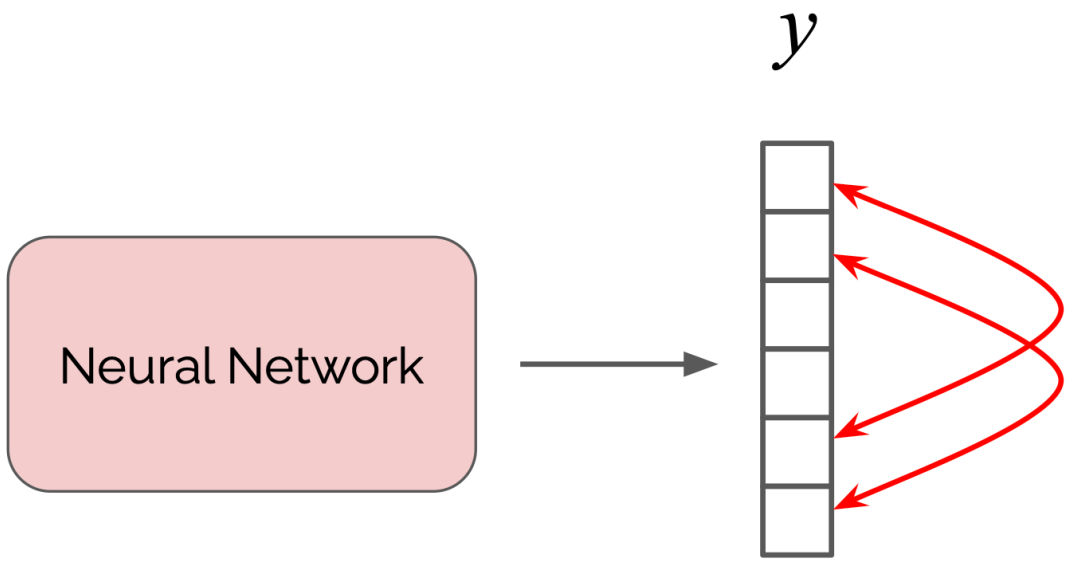
What is the Role of Logic and Reasoning



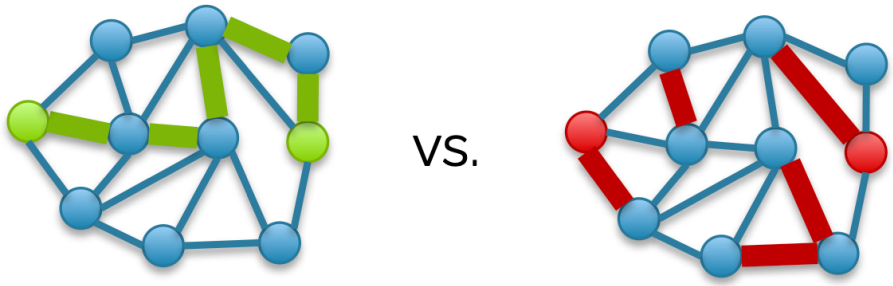
Monotonicity (Prior Knowledge):

Increasing income should increase the approved loan amount

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

Logic Representations

Syntax

- propositional variables
- logical connectives
  $\Rightarrow$ Boolean combinations

Syntax: Propositional Variables

Let Π be a set of *propositional variables*.

We use letters P, Q, R, S, to denote propositional variables.

Syntax: Formula

F_Π is the set of propositional formulas over Π defined inductively as follows:

$$
\begin{array}{rcll}
F, G & ::= & \bot & (\text{falsum}) \\
& | & \top & (\text{verum}) \\
& | & P, \quad P \in \Pi & (\text{atomic formula}) \\
& | & (\neg F) & (\text{negation}) \\
& | & (F \wedge G) & (\text{conjunction}) \\
& | & (F \vee G) & (\text{disjunction}) \\
& | & (F \to G) & (\text{implication})
\end{array}
$$

Syntax: Formula

F_Π is the set of propositional formulas over Π defined inductively as follows:

$$
\begin{array}{rcll}
F, G & ::= & \bot & \text{(falsum)} \\
& | & \top & \text{(verum)} \\
& | & P, \quad P \in \Pi & \text{(atomic formula)} \\
& | & (\neg F) & \text{(negation)} \\
& | & (F \wedge G) & \text{(conjunction)} \\
& | & (F \vee G) & \text{(disjunction)} \\
& | & (F \rightarrow G) & \text{(implication)}
\end{array}
$$

We only need negation, And, Or

Syntax: Formula

F_Π is the set of propositional formulas over Π defined inductively as follows:

$$
\begin{array}{rcll}
F, G & ::= & \bot & \text{(falsum)} \\
 & | & \top & \text{(verum)} \\
 & | & P, \quad P \in \Pi & \text{(atomic formula)} \\
 & | & (\neg F) & \text{(negation)} \\
 & | & (F \wedge G) & \text{(conjunction)} \\
 & | & (F \vee G) & \text{(disjunction)} \\
 & | & (F \to G) & \text{(implication)}
\end{array}
$$

How to understand implication?

SAT (Satisfiability) Reasoning

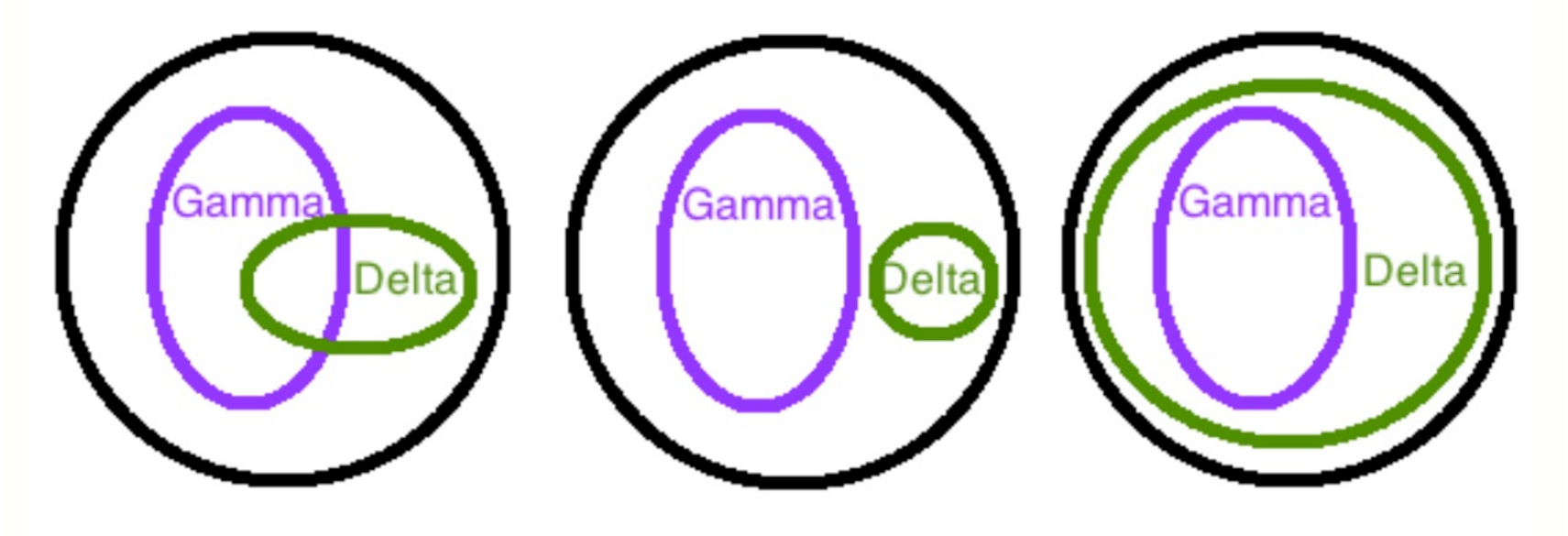
Reasoning Reduces to SAT

The basic inference problem in the propositional logic system is the following: given the knowledge that we have about the world, is it the case that some property must always hold?

In other word,

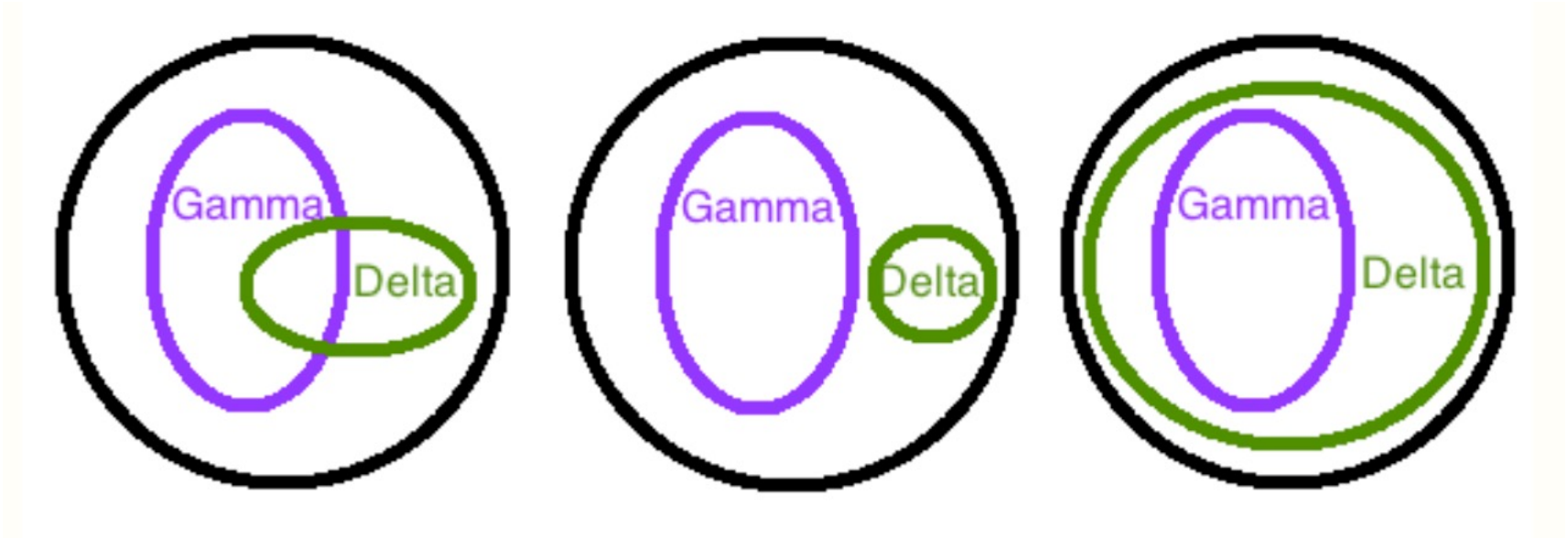
Does α imply β?

An Exercise: SAT



After adding Delta, how will the number of possible SAT worlds change?

An Exercise: SAT



(i) decrease; (ii) go to zero; (iii) remain the sames

An Example: Sudoku

	1	2	3	4	5	6	7	8	9
1								1	
2	4								
3		2							
4					5		4		7
5			8				3		
6			1		9				
7	3			4			2		
8		5		1					
9				8		6			

Goal:

Fill the empty fields with digits $1,\ldots,9$ so that each digit occurs exactly once in each row, column, and 3×3 box

Credit to: Dr. Uwe Waldmann

An Example: Sudoku

	1	2	3	4	5	6	7	8	9
1								1	
2	4								
3		2							
4					5		4		7
5			8				3		
6			1		9				
7	3			4			2		
8		5		1					
9				8		6			

Idea:

Use boolean variables $P^d_{i,j}$ with $d, i, j \in \{1, \ldots, 9\}$ to encode the problem:

$P^d_{i,j}$=true iff the value of square i, j is d

Credit to: Dr. Uwe Waldmann

An Example: Sudoku

	1	2	3	4	5	6	7	8	9
1								1	
2	4								
3		2							
4					5		4		7
5			8				3		
6			1		9				
7	3			4			2		
8		5		1					
9				8		6			

Idea:

Use boolean variables $P^d_{i,j}$ with $d, i, j \in \{1, \ldots, 9\}$ to encode the problem:

$P^d_{i,j}$=true iff the value of square i, j is d

For example:

$P^8_{5,3} = true$

Credit to: Dr. Uwe Waldmann

An Example: Sudoku

- Concrete values result in formulas $P^d_{i,j}$
- For every square (i, j) we generate $P^1_{i,j} \vee \ldots \vee P^9_{i,j}$
- For every square (i, j) and pair of values $d < d'$ we generate $\neg P^d_{i,j} \vee \neg P^{d'}_{i,j}$
- For every value d and row i we generate $P^d_{i,1} \vee \ldots \vee P^d_{i,9}$
  (Analogously for columns and 3×3 boxes)
- For every value d, row i, and pair of columns $j < j'$
  we generate $\neg P^d_{i,j} \vee \neg P^d_{i,j'}$,
  (Analogously for columns and 3×3 boxes)

An Example: Sudoku

Every assignment of boolean values to the variables $P^d_{i,j}$
so that all formulas become true
corresponds to a Sudoku solution (and vice versa).

An Example: Sudoku

Every assignment of boolean values to the variables $P^d_{i,j}$ so that all formulas become true corresponds to a Sudoku solution (and vice versa).

world	A	B	C
w_1	1	1	1
w_2	1	1	0
w_3	1	0	1
w_4	1	0	0
w_5	0	1	1
w_6	0	1	0
w_7	0	0	1
w_8	0	0	0

Brute force?

An Example: Sudoku

Every assignment of boolean values to the variables $P^d_{i,j}$
so that all formulas become true
corresponds to a Sudoku solution (and vice versa).

world	A	B	C
w_1	1	1	1
w_2	1	1	0
w_3	1	0	1
w_4	1	0	0
w_5	0	1	1
w_6	0	1	0
w_7	0	0	1
w_8	0	0	0

Niklas Eén, Niklas Sörensson:
MiniSat (`http://minisat.se/`)

An Example: Sudoku

world	A	B	C
w_1	1	1	1
w_2	1	1	0
w_3	1	0	1
w_4	1	0	0
w_5	0	1	1
w_6	0	1	0
w_7	0	0	1
w_8	0	0	0

Niklas Eén, Niklas Sörensson:

MiniSat (`http://minisat.se/`)

Unfortunately,

The satisfiability problem is NP-complete.

Every known algorithm to solve it has an exponential time worst-case behaviour (or worse).

SAT in Practice

- Hardware/software verification
- Dependency checking
- Programming language compilation
- Design automation
- Combinatorial problem solvers
- Theorem proving
- …

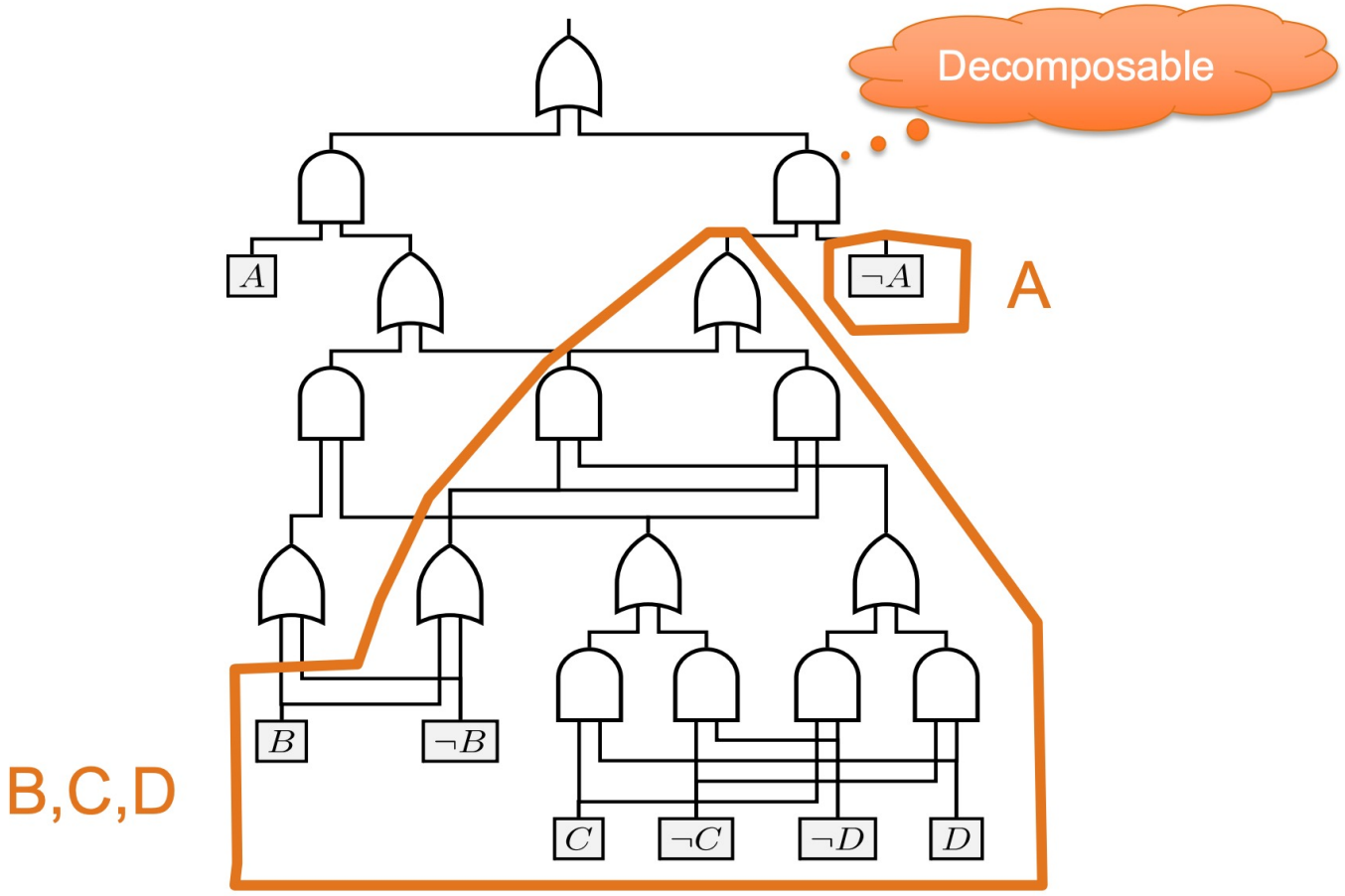
Logic Representation in Terms of SAT

Representation for Easy SAT?

- Is there a solution? (SAT)
  - SAT($\alpha \vee \beta$) iff SAT(α) or SAT(β) (*always*)
  - SAT($\alpha \wedge \beta$) iff ***???***

Darwiche and Marquis, "A knowledge compilation map", 2002

Representation: Decomposability



Disjoint, non-overlapping variable partition

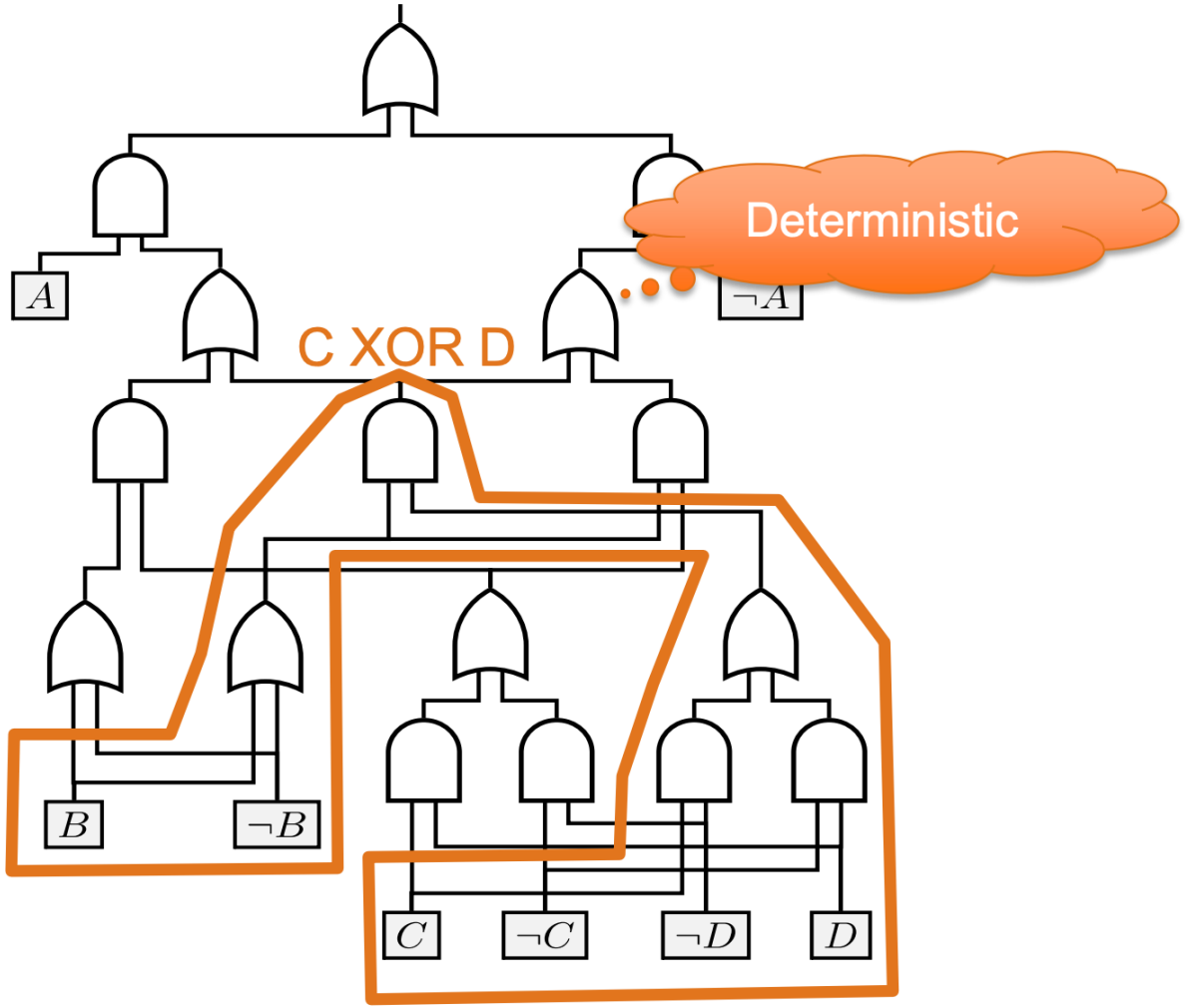
Darwiche and Marquis, "A knowledge compilation map", 2002

Representation: #SAT

- Is there a solution? (SAT) ✓
  - SAT($\alpha \vee \beta$) iff SAT(α) or SAT(β) (*always*)
  - SAT($\alpha \wedge \beta$) iff SAT(α) and SAT(β) (*decomposable*)
- How many solutions are there? (#SAT)

Darwiche and Marquis, "A knowledge compilation map", 2002

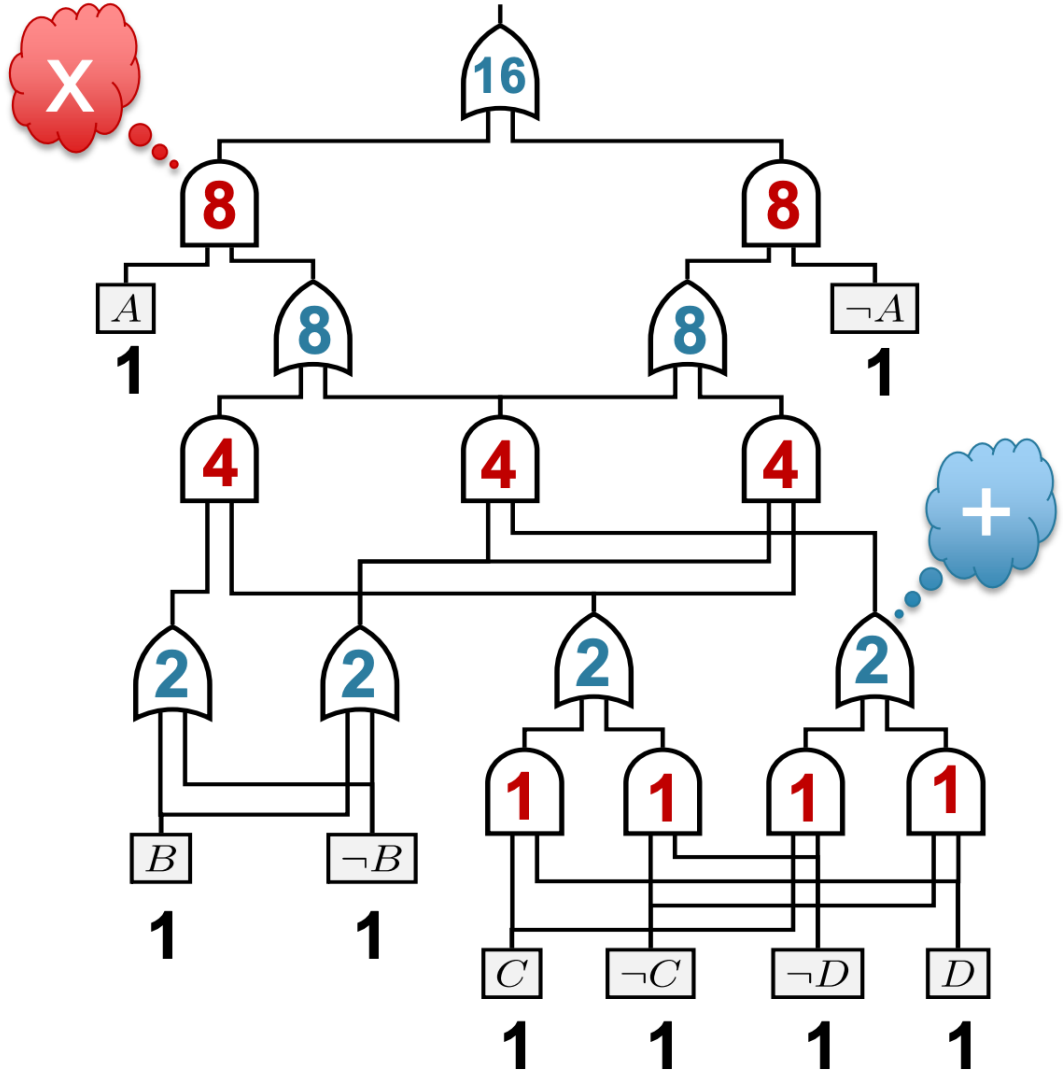
Representation: Determinism



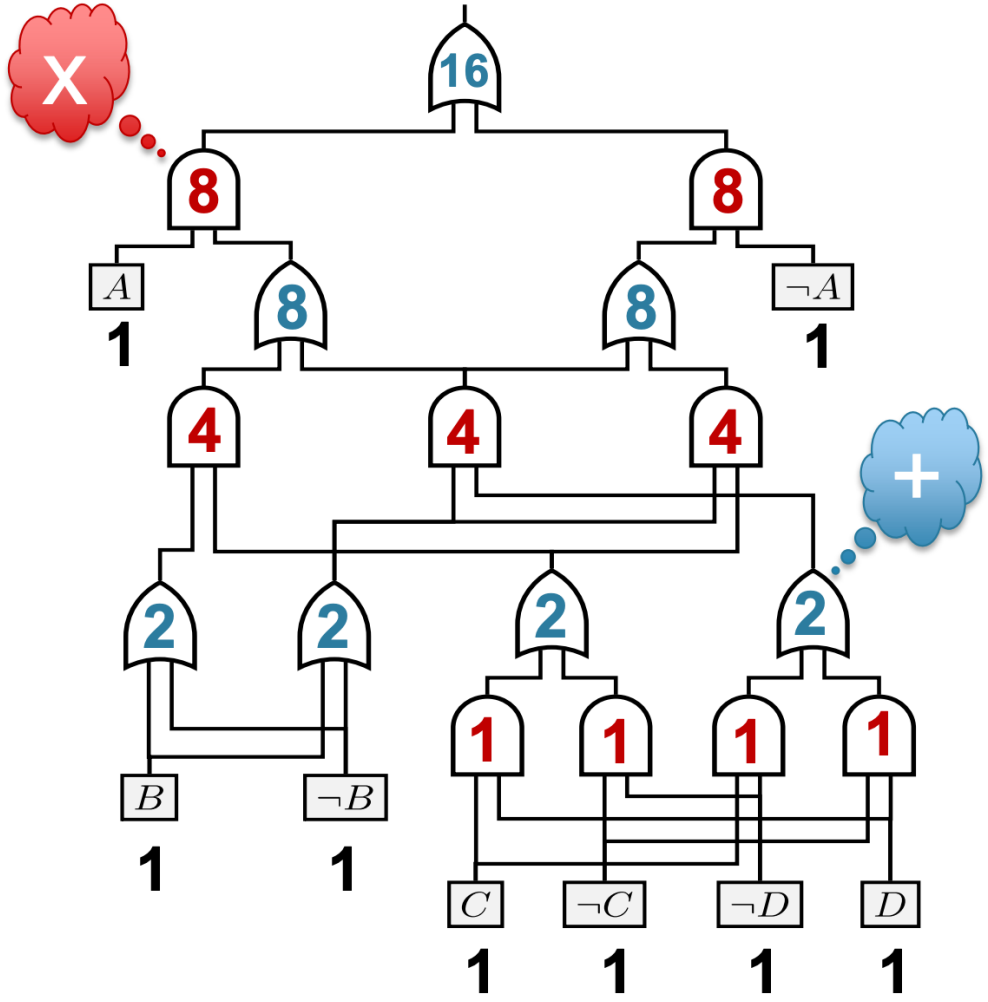
Mutually exclusive, exhaustive assignment partition

Darwiche and Marquis, "A knowledge compilation map", 2002

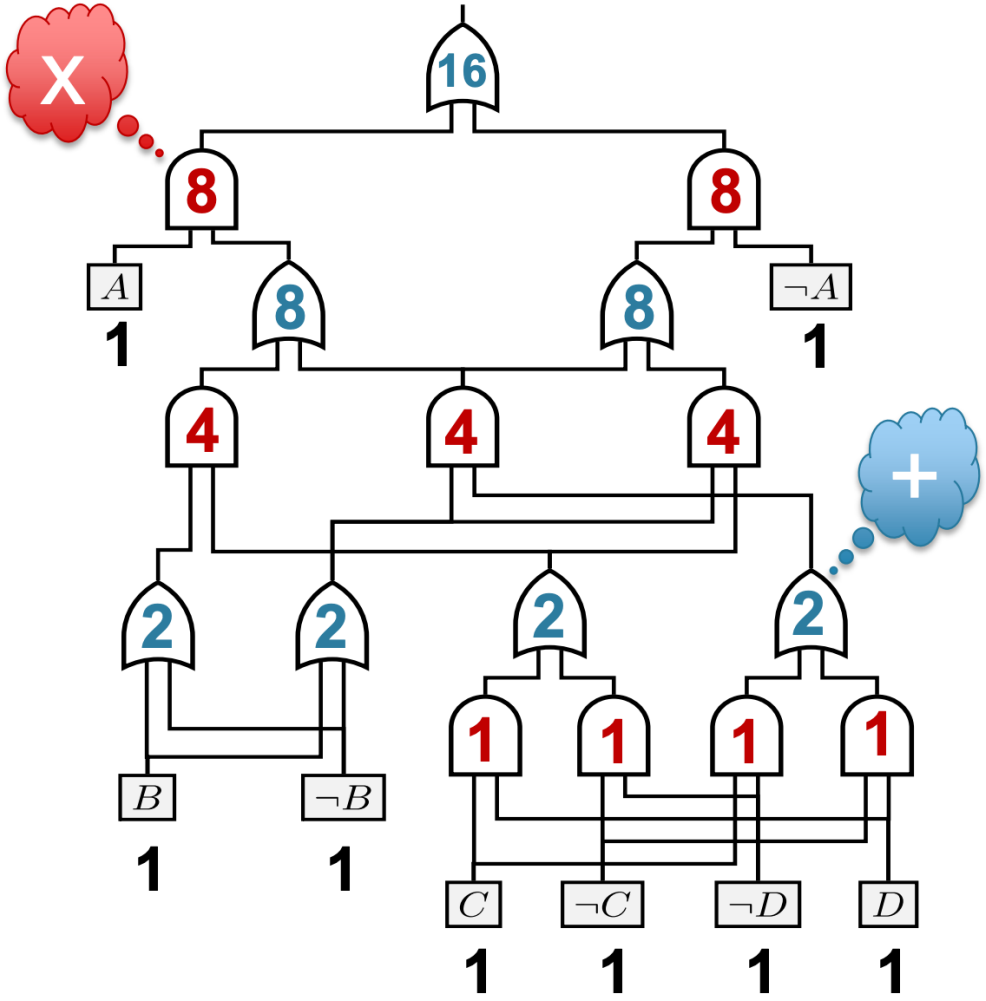
Representation: #SAT



Exercise: #SAT -> A Solution



Exercise: #SAT -> A Solution



Following a path that is non-zero

Representation: Tractable Circuits

- Is there a solution? (SAT) ✓
- How many solutions are there? (#SAT) ✓

- Complexity linear in circuit size ☺

Representation: Some Insights

- The circuit can be exponential in size of inputs
- Compiling this circuit is NP-Hard
- Then what is the benefit?

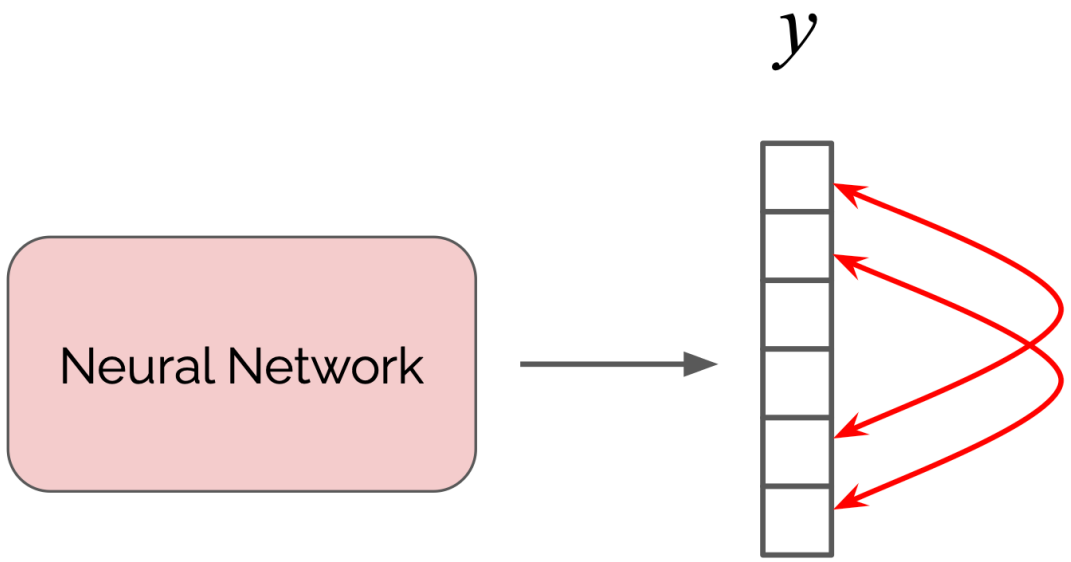
Representation: Some Insights

- The circuit can be exponential in size of inputs
- Compiling this circuit is NP-Hard
- Then what is the benefit?

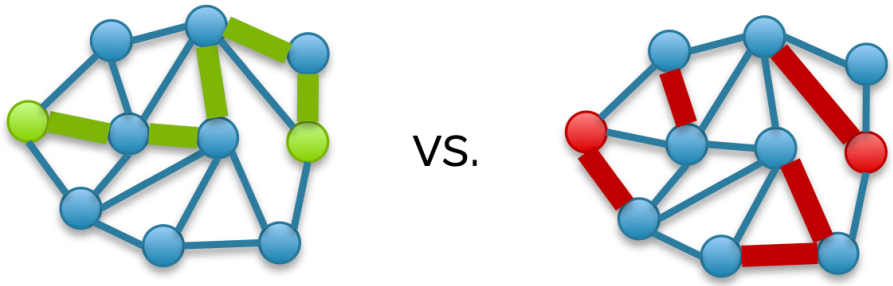
Amortization (only pay the exponential price once)

Neural Symbolic Learning

Recall: Declarative Knowledge on Outputs



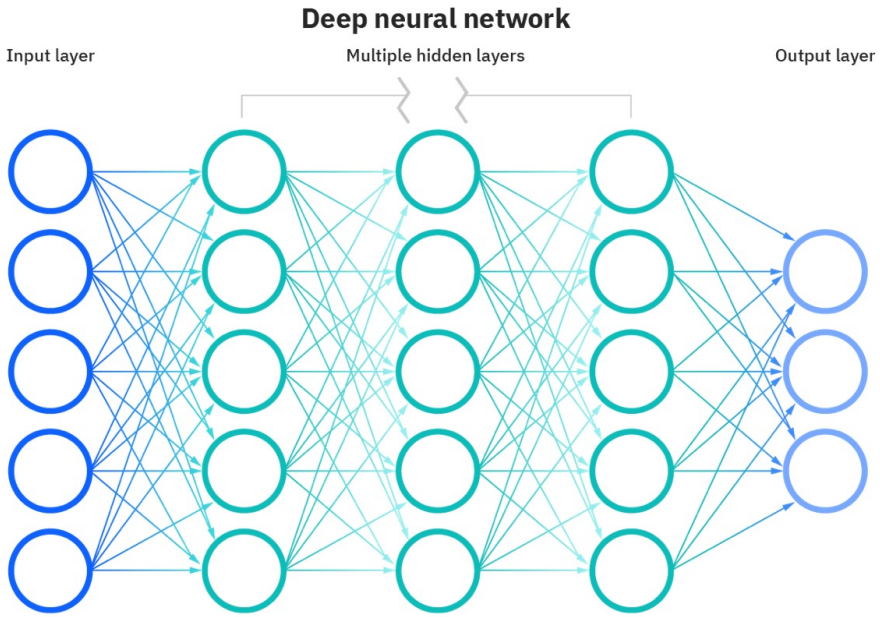
How is the output structured?
Are all possible outputs valid?



How are the outputs related to each other?

Learning this from data is inefficient
Much easier to express this declaratively

A Running Example: Classification



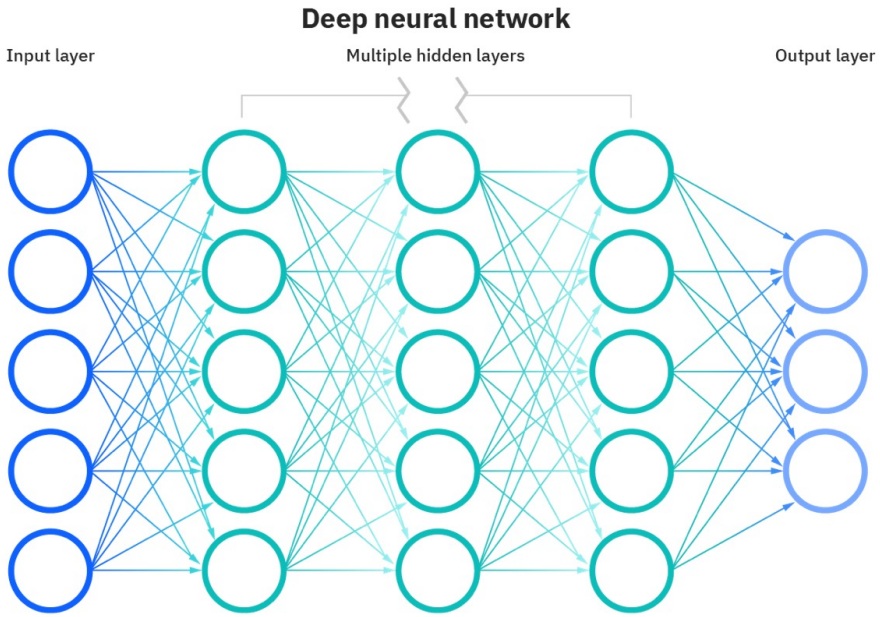
p

Learning subject to Exactly-One

$\neg Light \wedge Ball$
$\neg Ball \wedge Light$
Etc.

α

A Running Example: Classification



p

Learning subject to Exactly-One

$$\neg Light \wedge Ball$$
$$\neg Ball \wedge Light$$
Etc.

α

Output is probability vector p, not Boolean logic!

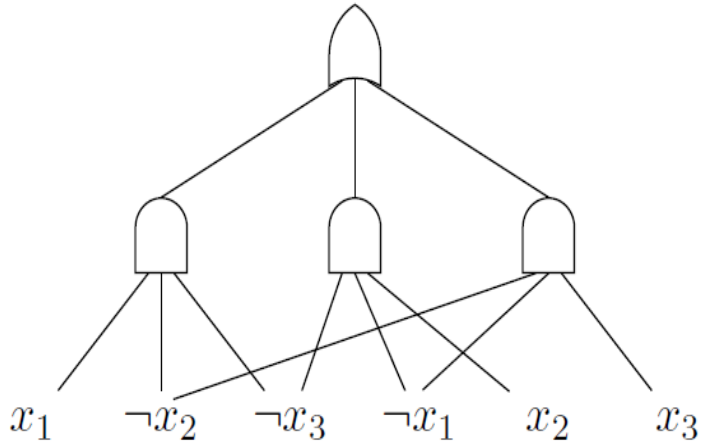
Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang and Guy Van den Broeck. A Semantic Loss Function for Deep Learning with Symbolic Knowledge, *ICML*, 2018.

Representation: Exactly One

Learning subject to Exactly-One

$\neg Light \wedge Ball$
$\neg Ball \wedge Light$
Etc.

α



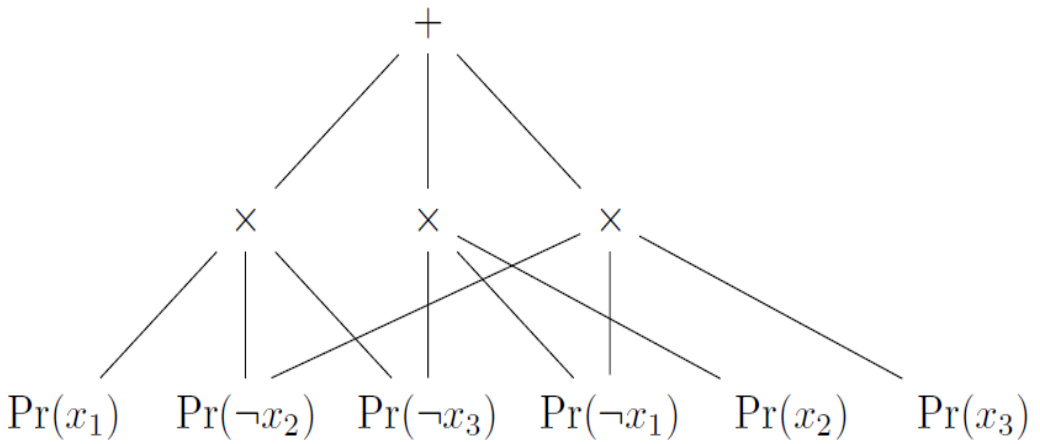
Representation: Prob Exactly One

Learning subject to Exactly-One

$\neg Light \wedge Ball$
$\neg Ball \wedge Light$
Etc.

α

$L(\alpha, \mathbf{p}) = - \log(\quad)$



Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang and Guy Van den Broeck. A Semantic Loss Function for Deep Learning with Symbolic Knowledge, *ICML*, 2018.

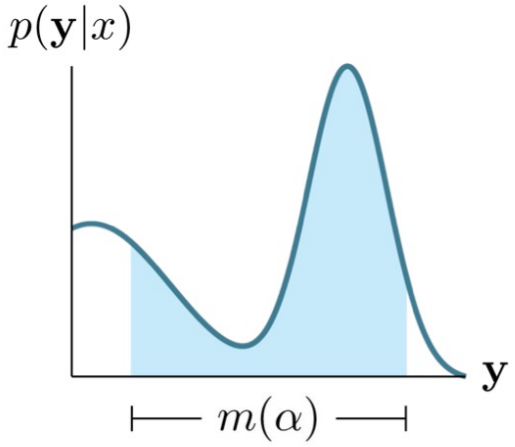
Semantic Loss

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

- Go over every model that satisfies (The circuit does it for us)
- Count the probability of every model

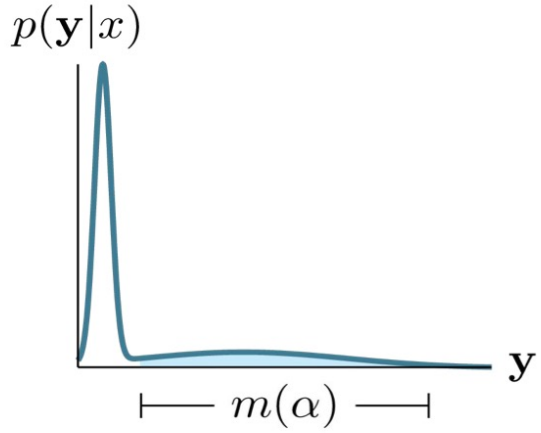
Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang and Guy Van den Broeck. A Semantic Loss Function for Deep Learning with Symbolic Knowledge, *ICML*, 2018.

Semantic Loss: Effects

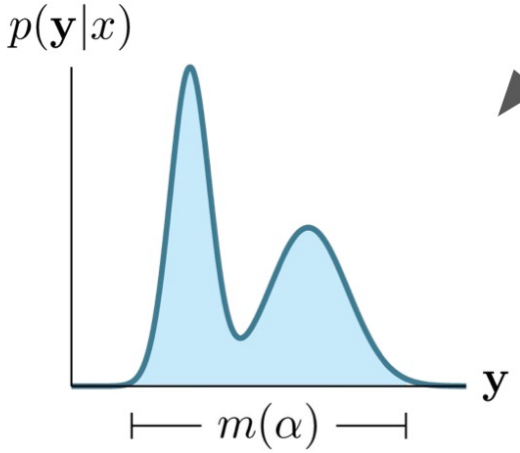


a) A network uncertain over both valid & invalid predictions

Semantic Loss



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

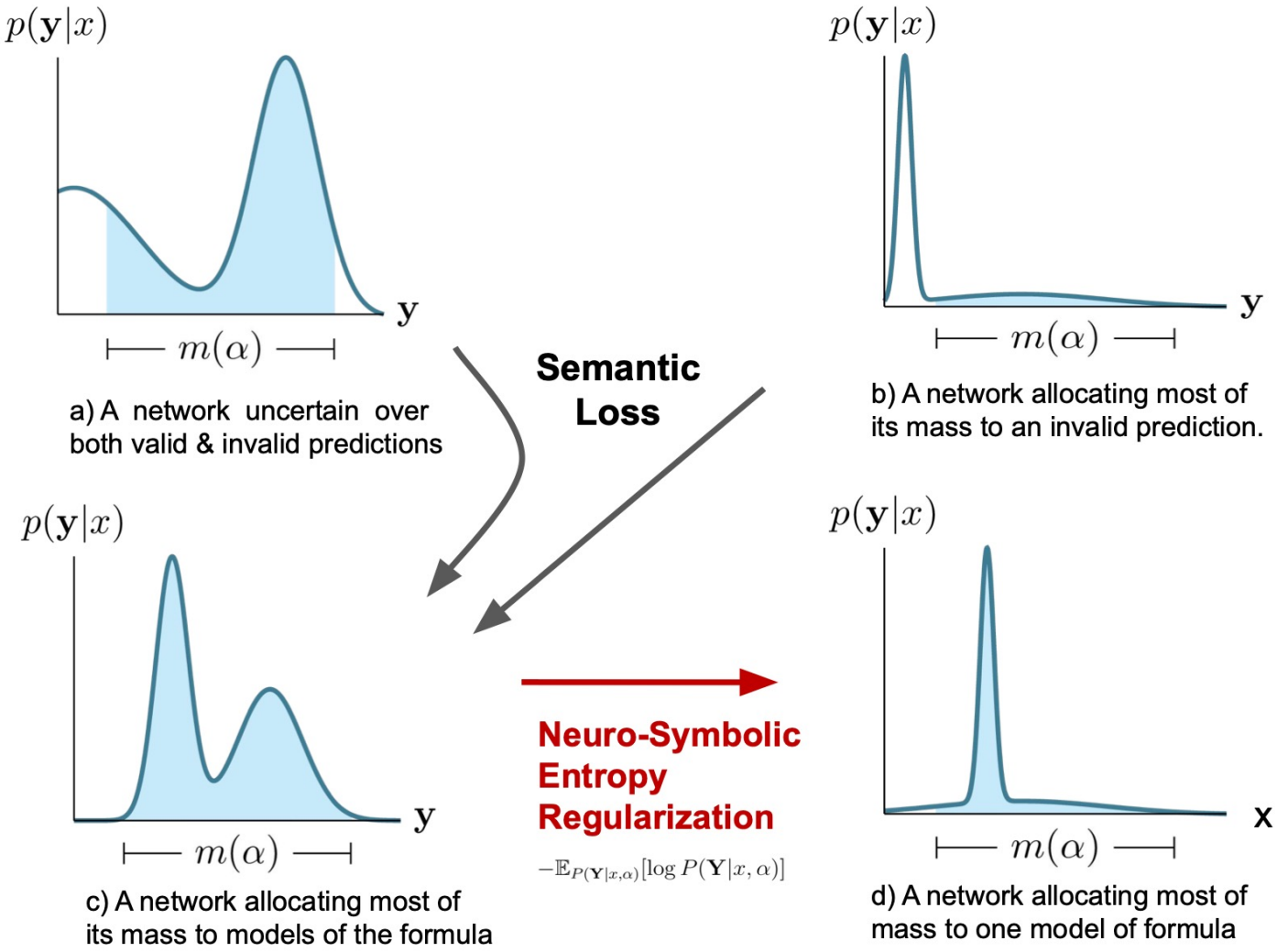


c) A network allocating most of its mass to models of the formula

$$P(\alpha|x) \uparrow : -\log P(\alpha|x) \downarrow$$

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang and Guy Van den Broeck. A Semantic Loss Function for Deep Learning with Symbolic Knowledge, *ICML*, 2018.

Semantic Loss + Regularization



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

Pylon

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
  - Monte-carlo estimate of loss
  - T-norm approximation
  - *your solver?*

http://pylon-lib.github.io

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
```

http://pylon-lib.github.io

Pylon

1 Specify knowledge as a predicate

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

2 Add as loss to training

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

PyTorch Code

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

http://pylon-lib.github.io

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)

http://pylon-lib.github.io

An Exercise: XOR

PyTorch Code

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

1 Specify knowledge as a predicate

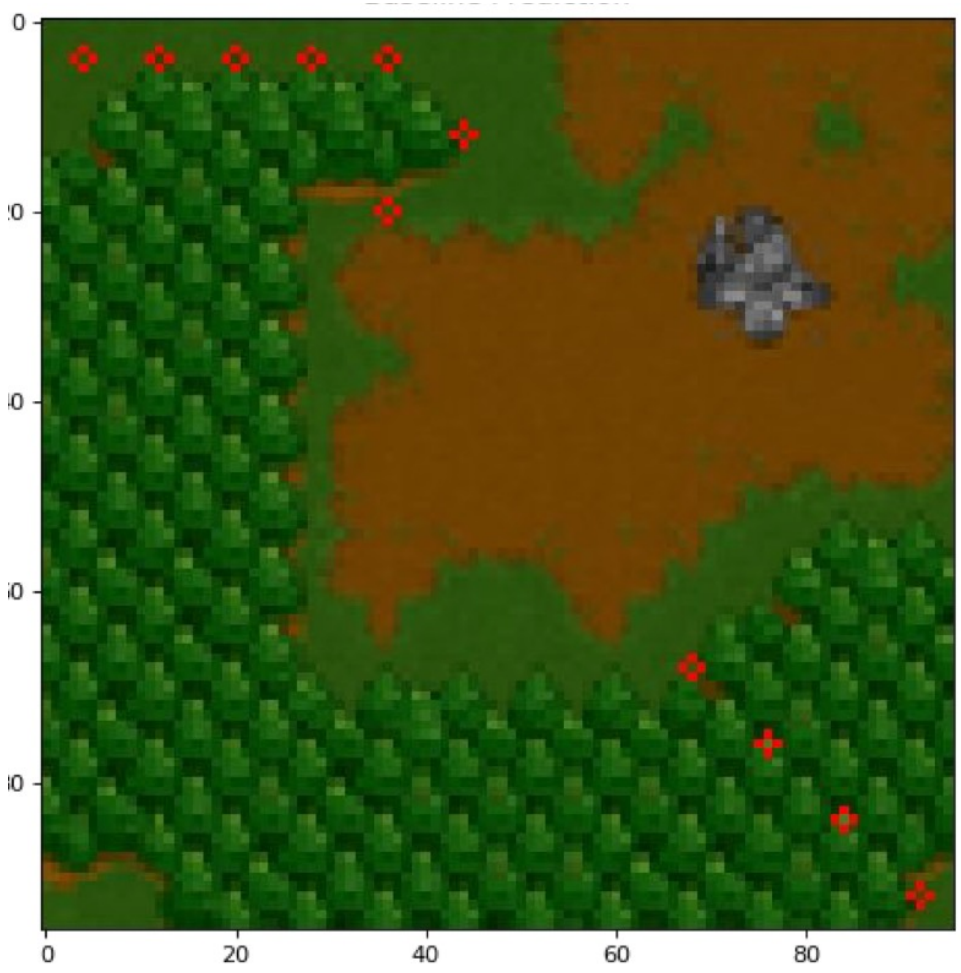
```
def check(y):
    ...
    return isValid
```

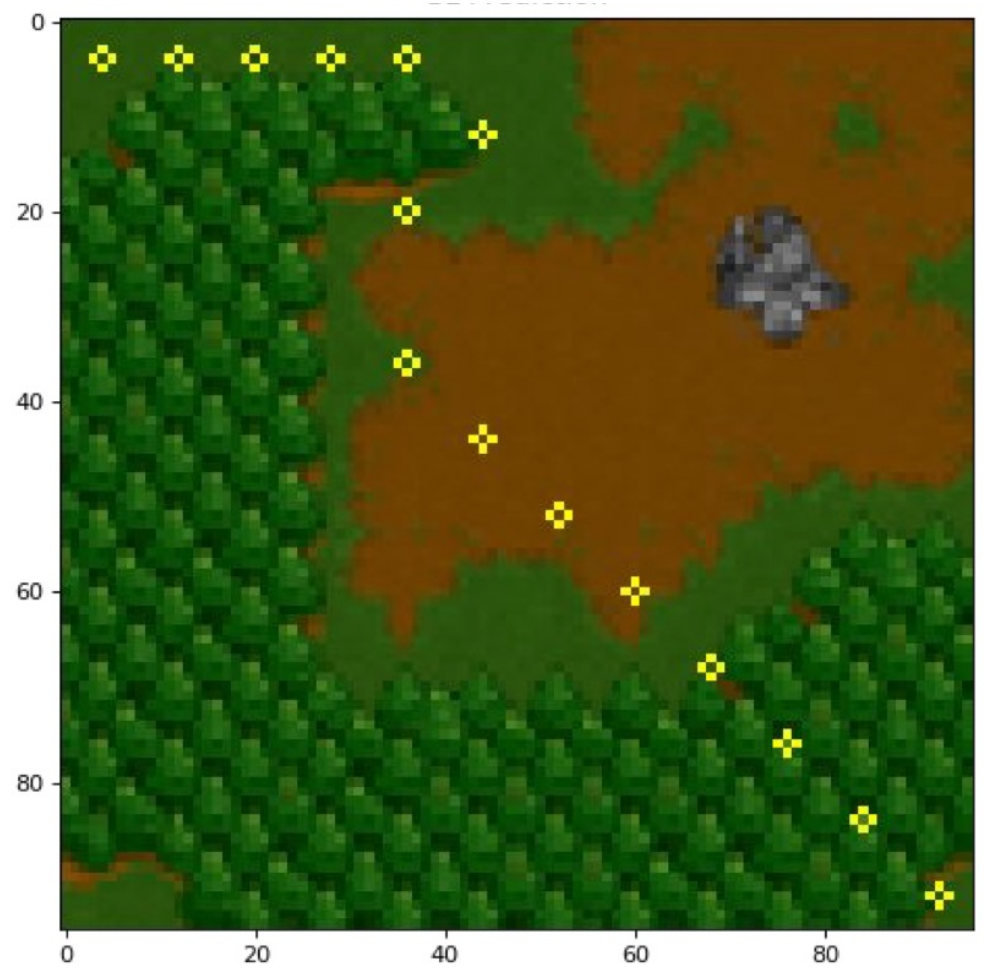
```
from pylon.constraint import constraint
from pylon.brute_force_solver import SatisfactionBruteForceSolver

# Our constraint function accepts a decoding tensor of
# shape (batch_size, ...) and is expected to return
# a tensor fo shape (batch_size, )
def xor(y):
    return y[:, 0] != y[:, 1] and y[:, 1] != y[:, 2]

xor_cons = constraint(xor, SatisfactionBruteForceSolver())
```

How to extend to exactly one?

http://pylon-lib.github.io

An Exercise: Path Constraint





http://pylon-lib.github.io

Reasoning with Programming

Task: Classify pairs of MNIST digits with their sum

https://github.com/ML-KULeuven/deepproblog

Reasoning with Programming

Task: Classify pairs of MNIST digits with their sum

Encode single-digit addition as logic

```
addition(X,Y,Z) :- digit(X,N1), digit(Y,N2), Z is N1+N2.
```

https://github.com/ML-KULeuven/deepproblog

Reasoning with Programming

Task: Classify pairs of MNIST digits with their sum

Put neural perception backs

```
nn(mnist_net, [X], Y, [0 ... 9] ) :: digit(X,Y).

addition(X,Y,Z) :- digit(X,N1), digit(Y,N2), Z is N1+N2.
```

https://github.com/ML-KULeuven/deepproblog

Reasoning with Programming

Task: Classify pairs of MNIST digits with their sum

Benefit: separate logic and knowledge

Put neural perception backs

```
nn(mnist_net, [X], Y, [0 ... 9] ) :: digit(X,Y).

addition(X,Y,Z) :- digit(X,N1), digit(Y,N2), Z is N1+N2.
```

https://github.com/ML-KULeuven/deepproblog

An Exercise: Multi-digit Addition

Learn to classify the sum of pairs of MNIST digits

Individual digits are not labeled!

E.g. ( ,  , 8)

Could be done by a CNN: classify the concatenation of both images into 19 classes

However: 

https://github.com/ML-KULeuven/deepproblog

An Exercise: Multi-digit Addition

Represent multi-digit number [H | T]

A recursive program

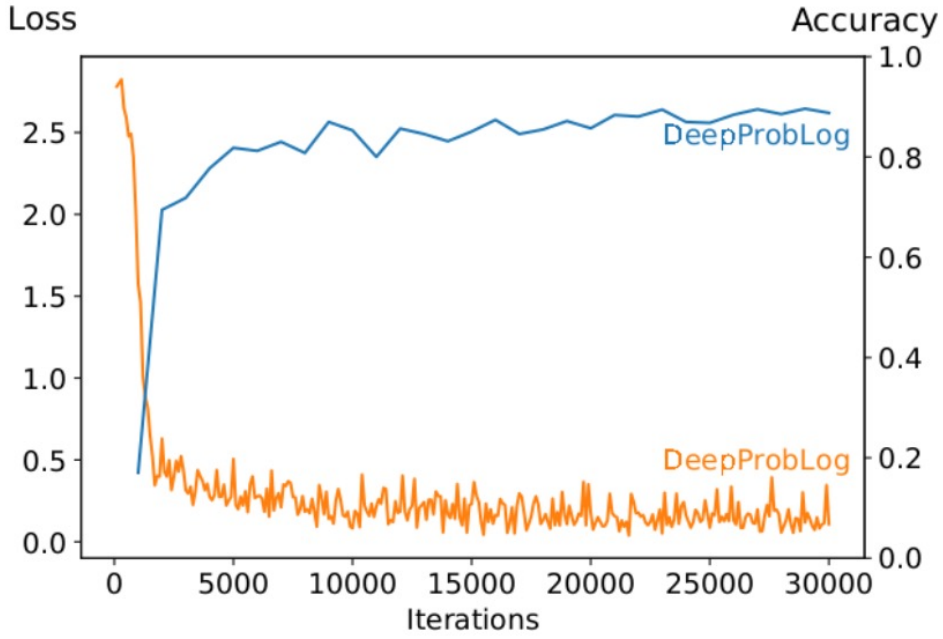
https://github.com/ML-KULeuven/deepproblog

An Exercise: Multi-digit Addition

Represent multi-digit number [H | T]

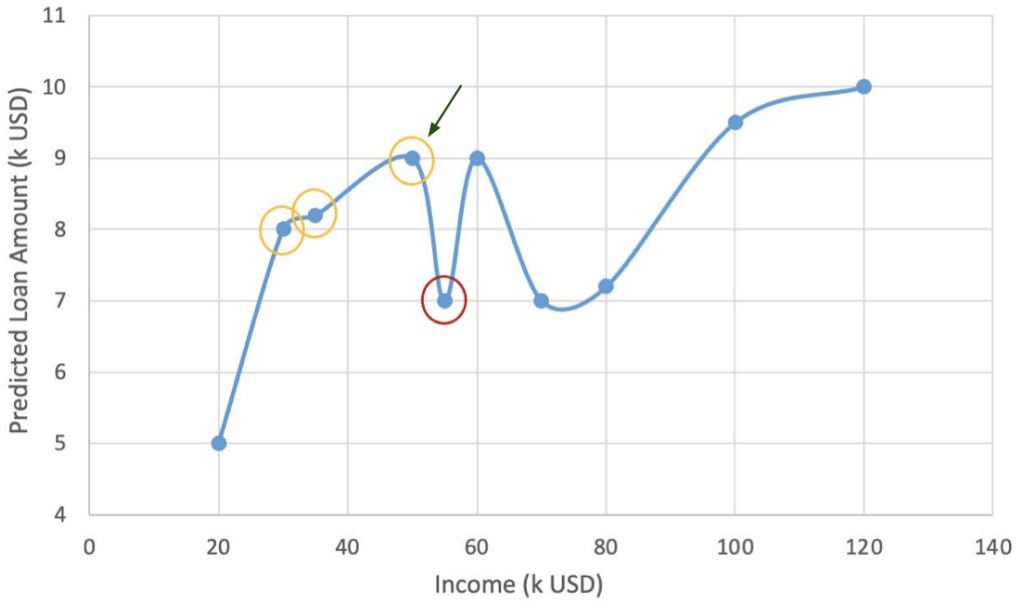
A recursive program

```
number ( [ ] , Result , Result ) .
number ( [H | T ] , Acc , Result) :-
  digit(H, Nr ), Acc2 is Nr +10*Acc ,
  number ( T , Acc2 , Result ) .
number (X,Y) :- number (X, 0 ,Y ) .

multiaddition(X, Y, Z ) :-
    number (X, X2 ) ,
    number (Y, Y2 ) ,
    Z is X2+Y2 .
```



https://github.com/ML-KULeuven/deepproblog

Reasoning with Counter-Examples



$$\exists x, y \; x \leq y \implies f(x) > f(y)$$

Computed using SMT(LRA) logical reasoning solver

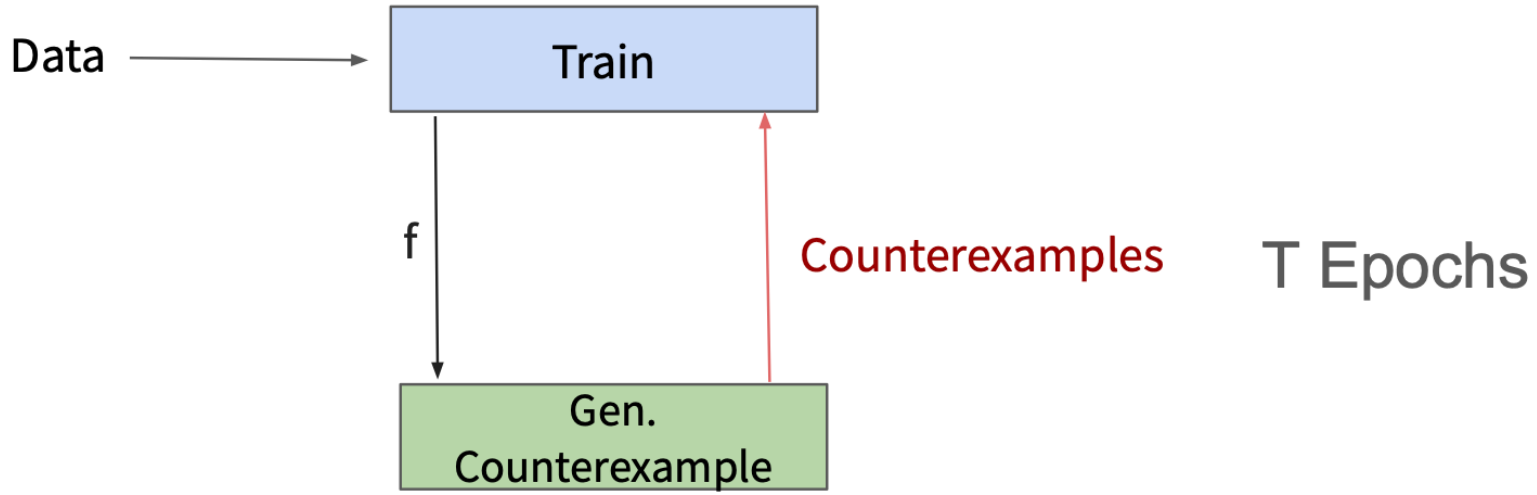
Maximal counterexamples (largest violation) using OMT

Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein and Guy Van den Broeck. Counterexample-Guided Learning of Monotonic Neural Networks, *NeurIPS*, 2020.

Reasoning with Counter-Examples

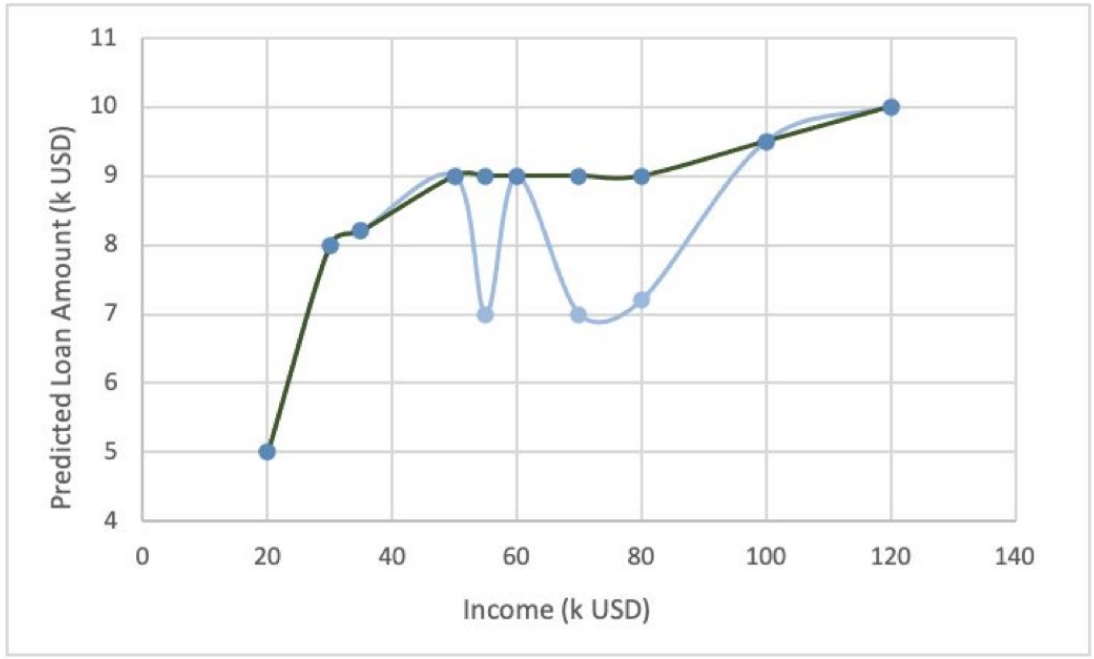
How to use monotonicity to improve model quality?

"Monotonicity as inductive bias"



Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein and Guy Van den Broeck. Counterexample-Guided Learning of Monotonic Neural Networks, *NeurIPS*, 2020.

Reasoning with Counter-Examples



Monotonic Envelope:

- Replace each prediction by its maximal counterexample
- Envelope construction is online (during prediction)
- Guarantees monotonic predictions for any ReLU neural net

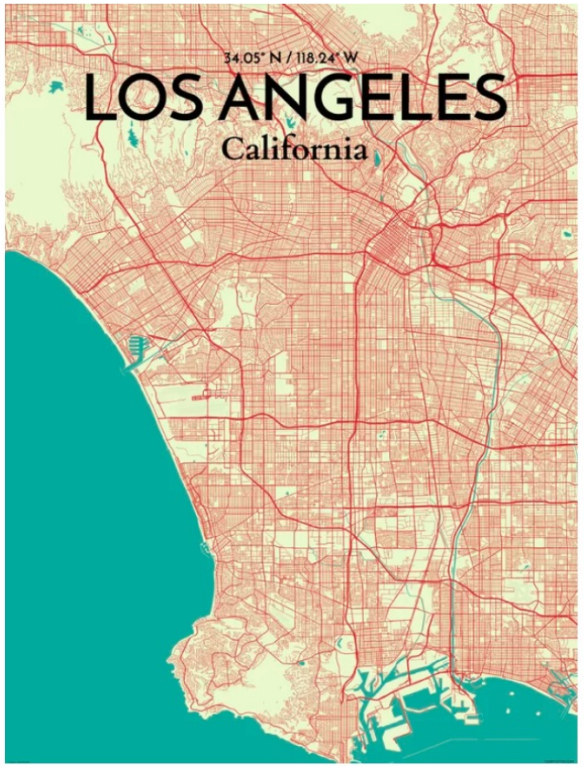
- Works for high-dimensional input
- Works for multiple monotonic features

Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein and Guy Van den Broeck. Counterexample-Guided Learning of Monotonic Neural Networks, *NeurIPS*, 2020.

Probabilistic Reasoning

Why Probabilistic Reasoning

q_1: *What is the probability that today is a Monday and there is a traffic jam on Westwood Blvd.?*

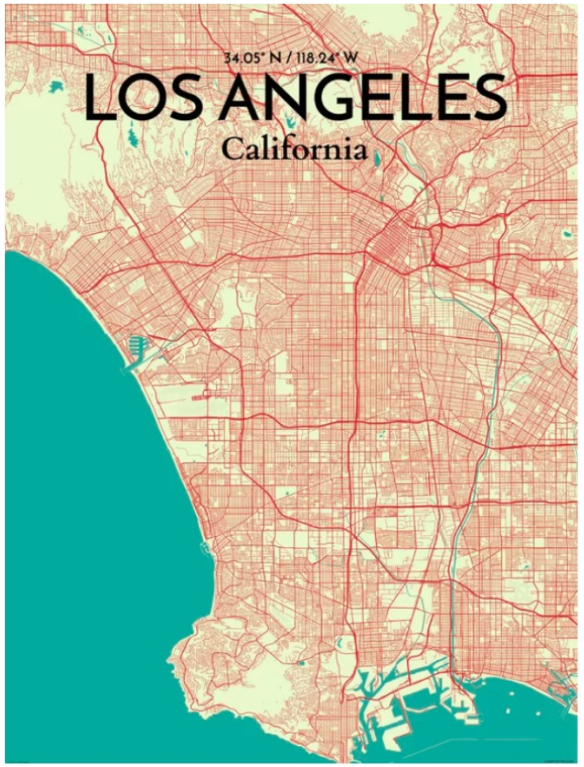


© fineartamerica.com

Why Probabilistic Reasoning

q_1: *What is the probability that today is a Monday and there is a traffic jam on Westwood Blvd.?*

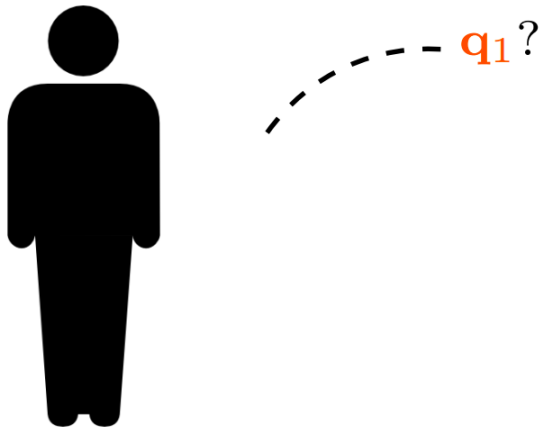
q_2: *Which day is most likely to have a traffic jam on my route to campus?*



© fineartamerica.com

Alternatives

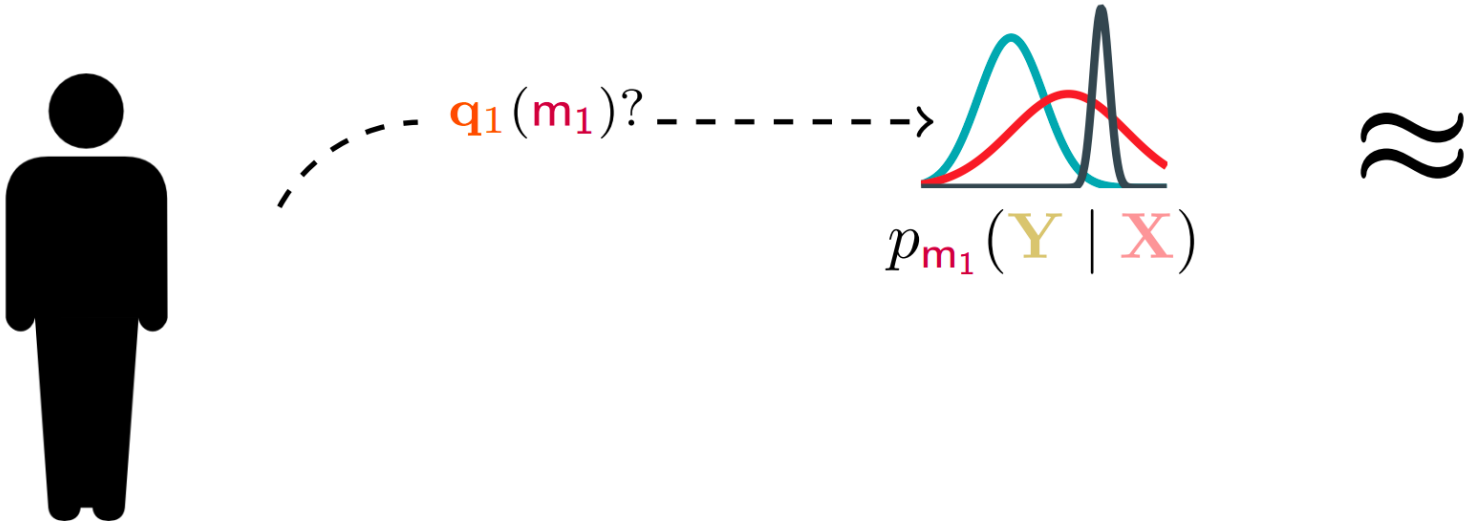
"What is the most likely street to have a traffic jam at 12.00?"



answering queries...

Alternatives

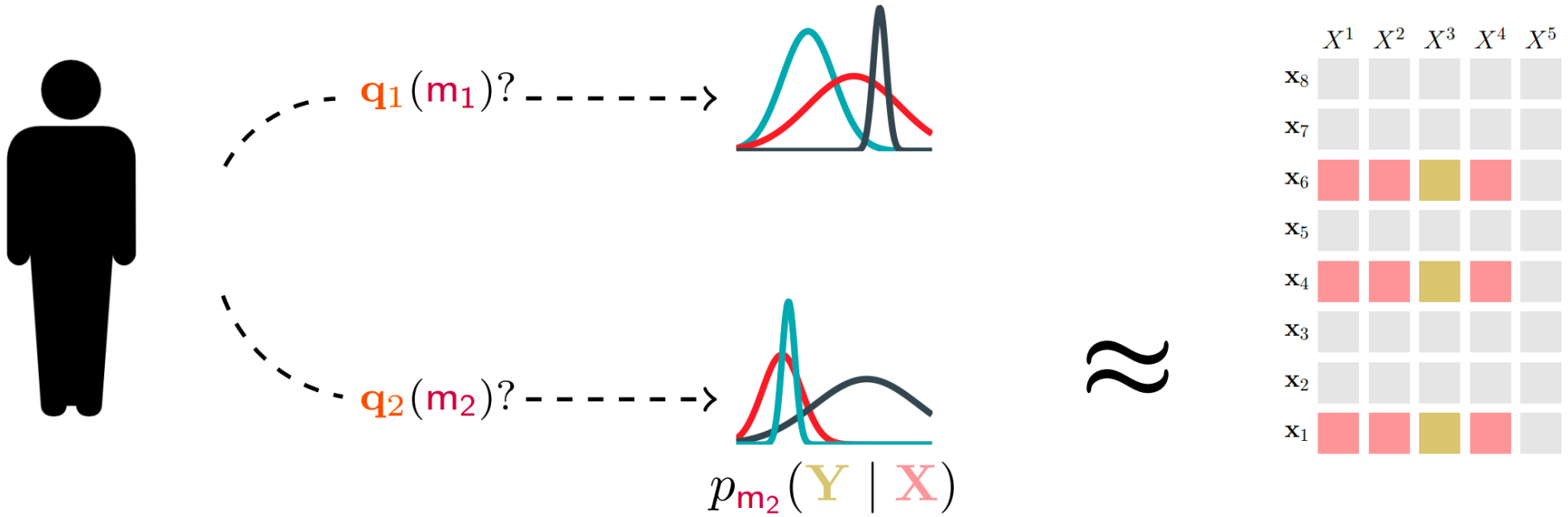
*"What is the most likely **street** to have a traffic jam at **12.00**?"*



...by fitting predictive models!

Alternatives

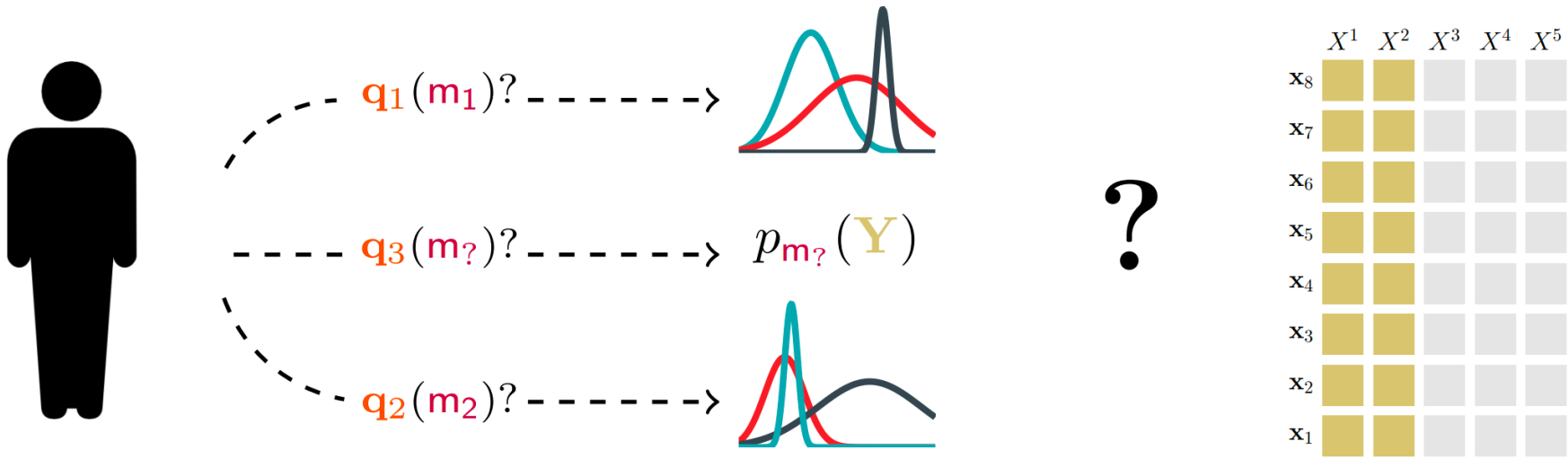
*"What is the most likely **time** to see a traffic jam at **Sunset Blvd.**?"*



~~***...by fitting predictive models!***~~

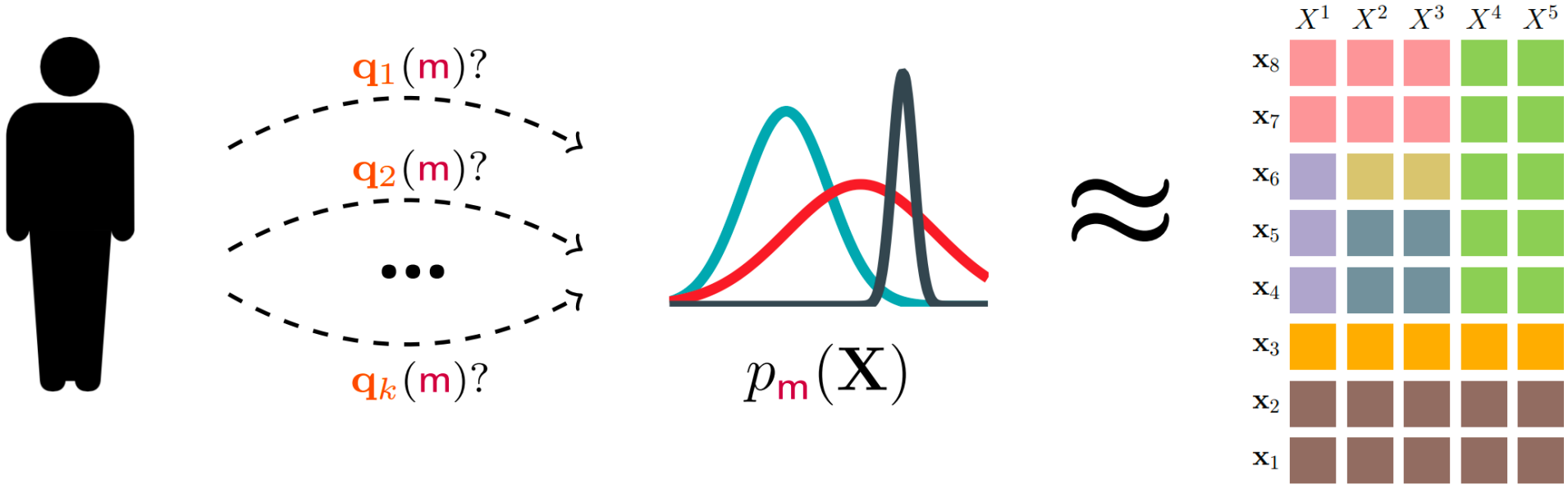
Alternatives

"What is the probability of a traffic jam on ***Westwood Blvd.*** *on* ***Monday****?"*



~~***...by fitting predictive models!***~~

Tractable Probabilistic Reasoning



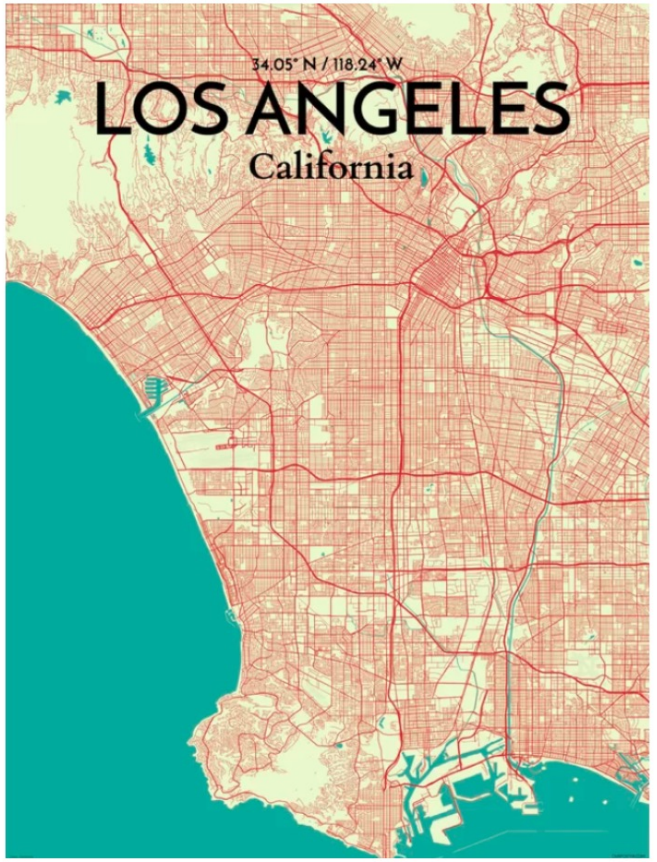
...by fitting generative models!

Probabilistic Reasoning: Complete Evidence

$\mathbf{q}_3$: *What is the probability that today is a Monday at 12.00 and there is a traffic jam only on Westwood Blvd.?*

$$\mathbf{X} = \{\text{Day}, \text{Time}, \text{Jam}_{\text{Wwood}}, \text{Jam}_{\text{Str2}}, \ldots, \text{Jam}_{\text{StrN}}\}$$

$$\mathbf{q}_3(\mathbf{m}) = p_{\mathbf{m}}(\mathbf{X} = \{\text{Mon}, 12.00, 1, 0, \ldots, 0\})$$



© fineartamerica.com

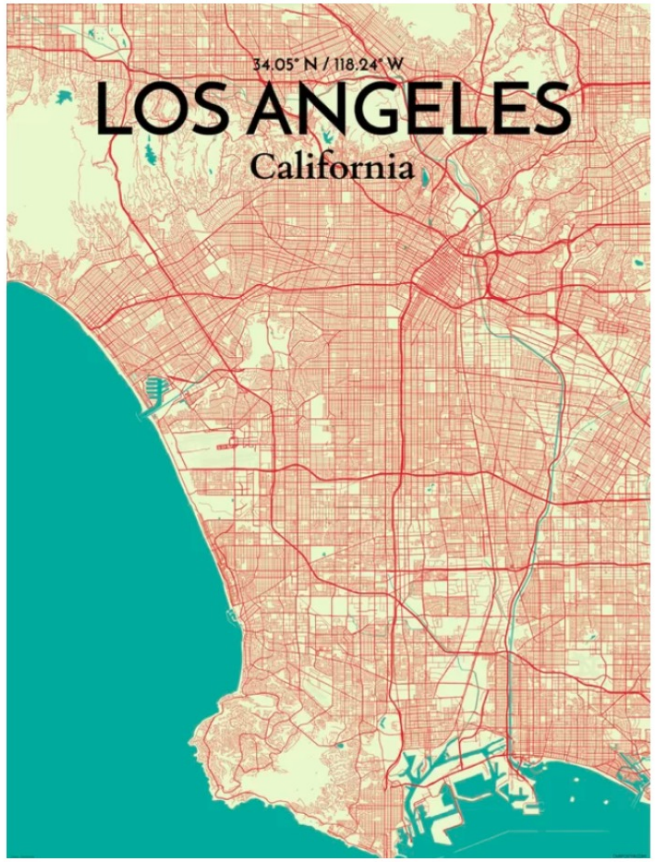
Probabilistic Reasoning: Marginals

$\mathbf{q}_1$: *What is the probability that today is a Monday ~~at 12.00~~ and there is a traffic jam ~~only~~ on Westwood Blvd.?*

$$\mathbf{q}_1(\mathbf{m}) = p_{\mathbf{m}}(\text{Day} = \text{Mon}, \text{Jam}_{\text{Wwood}} = 1)$$

General: $p_{\mathbf{m}}(\mathbf{e}) = \int p_{\mathbf{m}}(\mathbf{e}, \mathbf{H})\, d\mathbf{H}$

where $\mathbf{E} \subset \mathbf{X}, \quad \mathbf{H} = \mathbf{X} \setminus \mathbf{E}$



© fineartamerica.com

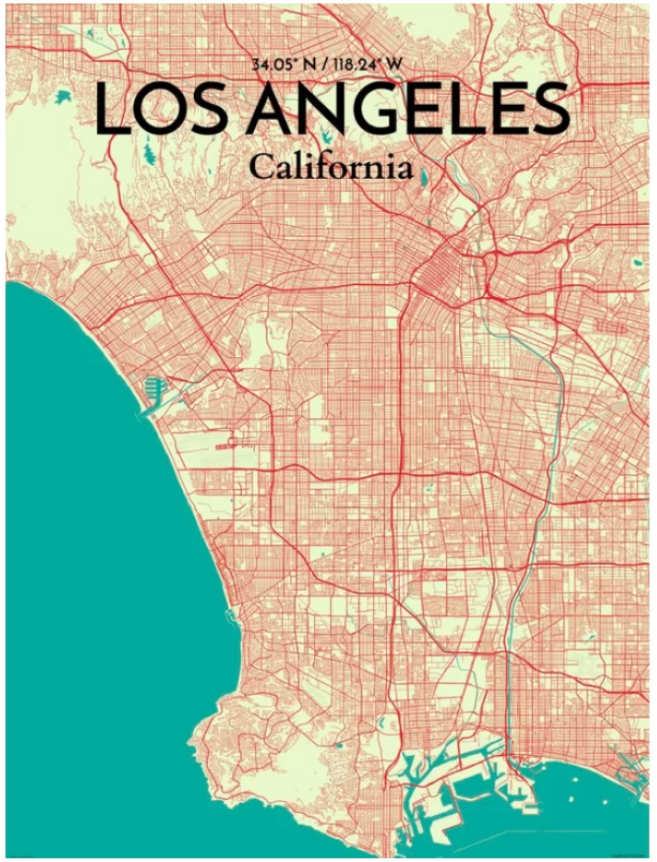
Probabilistic Reasoning: Marginals

$\mathbf{q}_1$: *What is the probability that today is a Monday ~~at 12.00~~ and there is a traffic jam ~~only~~ on Westwood Blvd.?*

$$\mathbf{q}_1(\mathbf{m}) = p_{\mathbf{m}}(\text{Day} = \text{Mon}, \text{Jam}_{\text{Wwood}} = 1)$$

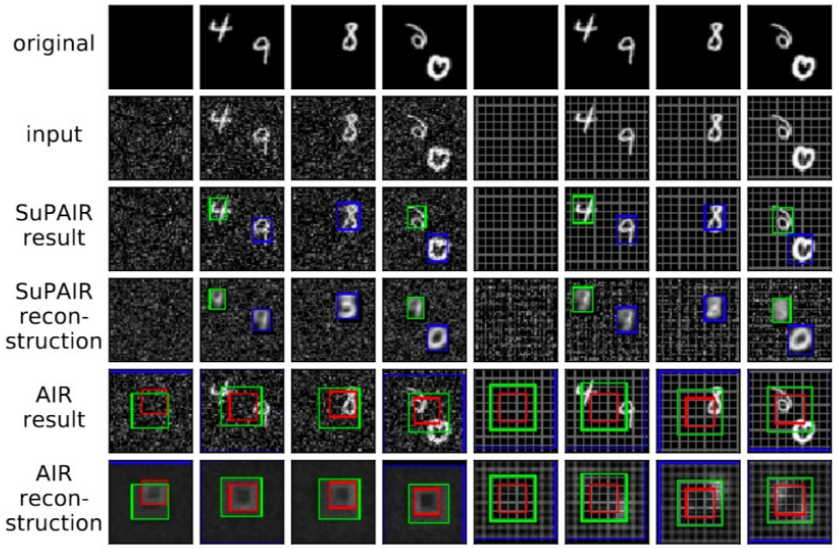
tractable MAR $\Longrightarrow$ tractable ***conditional queries*** (CON):

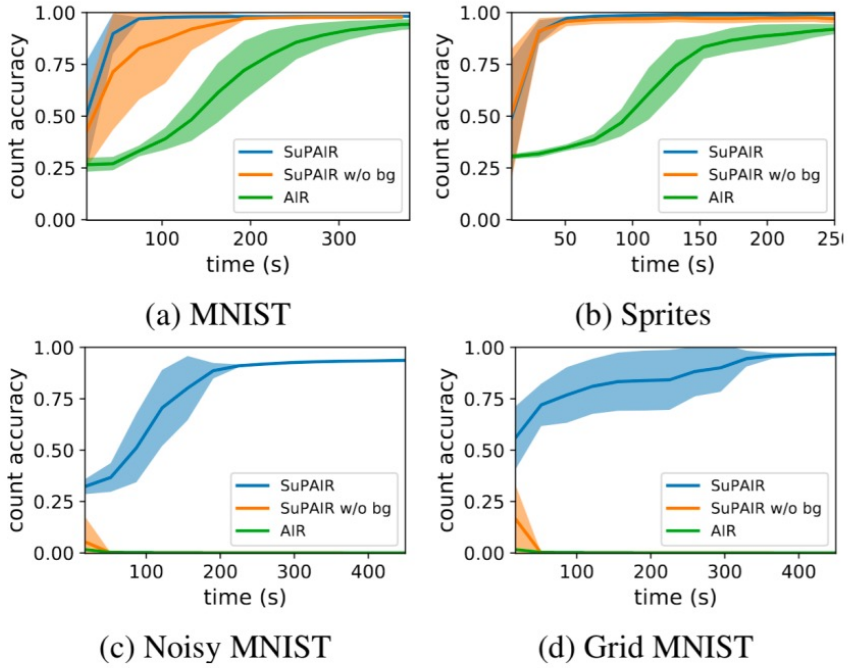
$$p_{\mathbf{m}}(\mathbf{q} \mid \mathbf{e}) = \frac{p_{\mathbf{m}}(\mathbf{q}, \mathbf{e})}{p_{\mathbf{m}}(\mathbf{e})}$$



© fineartamerica.com

Probabilistic Reasoning: Marginals





Fast and exact marginalization over unseen or “do not care” parts in the scene

Scene Understanding

Probabilistic Reasoning

A class of queries $\mathcal{Q}$ *is* tractable *on a family of probabilistic models* $\mathcal{M}$

iff for any query $\mathbf{q} \in \mathcal{Q}$ *and model* $\mathbf{m} \in \mathcal{M}$

exactly *computing* $\mathbf{q}(\mathbf{m})$ *runs in time* $O(\text{poly}(|\mathbf{m}|))$.

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

Probabilistic Reasoning

A class of queries $\mathcal{Q}$ *is* tractable *on a family of probabilistic models* $\mathcal{M}$

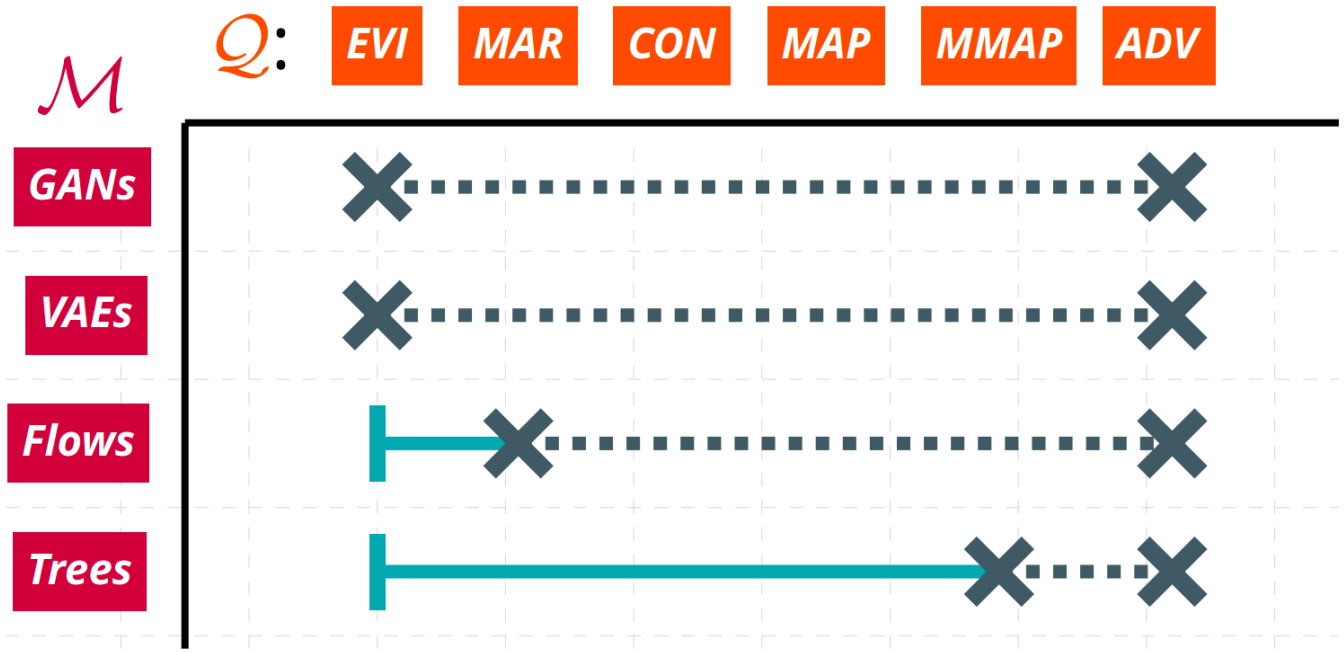
iff for any query $\mathbf{q} \in \mathcal{Q}$ *and model* $\mathbf{m} \in \mathcal{M}$

exactly *computing* $\mathbf{q}(\mathbf{m})$ *runs in time* $O(\text{poly}(|\mathbf{m}|))$.

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

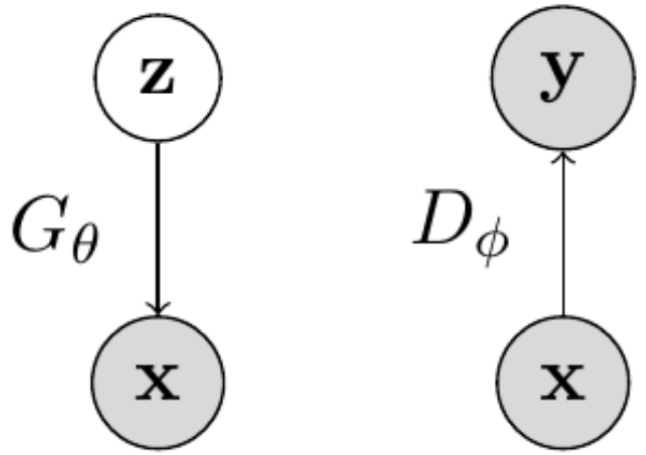
Weighted #SAT is a reasoning

Probabilistic Reasoning: A Spectrum



Probabilistic Reasoning: GANs?

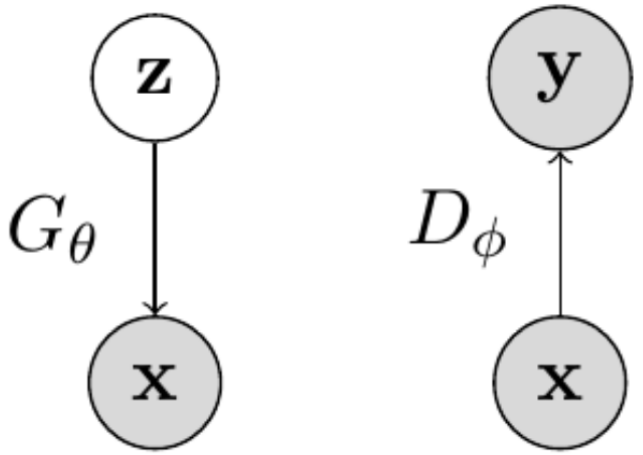
$$\min_\theta \max_\phi \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[\log D_\phi(\mathbf{x})\right] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log(1 - D_\phi(G_\theta(\mathbf{z})))\right]$$



Probabilistic Reasoning: GANs?

$$\min_\theta \max_\phi \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[\log D_\phi(\mathbf{x})\right] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log(1 - D_\phi(G_\theta(\mathbf{z})))\right]$$

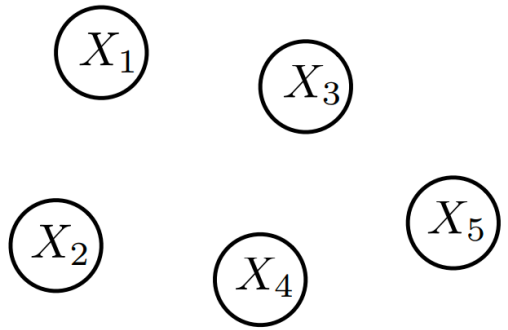
- no explicit likelihood!
  - $\Rightarrow$ *adversarial training instead of MLE*
    - $\Rightarrow$ *no tractable EVI*
- good sample quality
  - $\Rightarrow$ *but lots of samples needed for MC*
- unstable training $\Rightarrow$ *mode collapse*



What Have we Learned from Logical Reasonings

Full factorization: recall AND (decomposability)

A completely disconnected graph. Example: Product of Bernoullis (PoBs)



$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i)$$

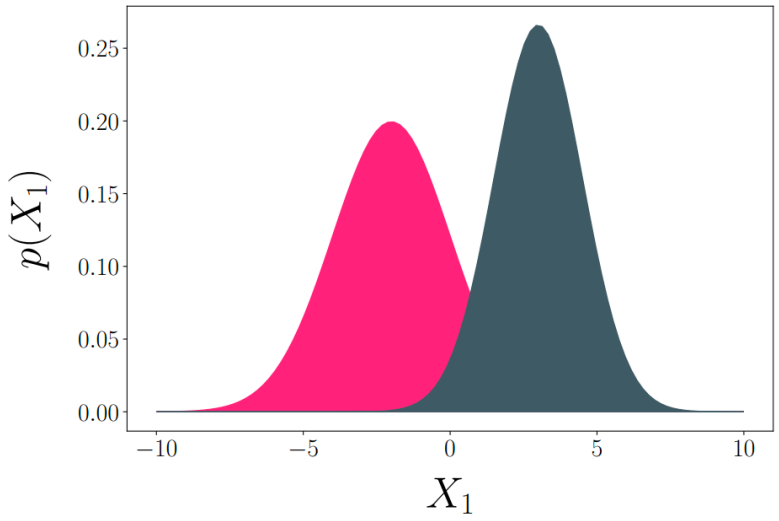
Complete evidence, marginals and MAP, MMAP inference is ***linear***!

$\Rightarrow$ *but definitely not expressive...*

What Have we Learned from Logical Reasonings

Mixtures: recall OR (soft determinism)

Mixtures as a convex combination of k (simpler) probabilistic models



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

Mixtures are marginalizing a ***categorical latent variable*** Z with k values

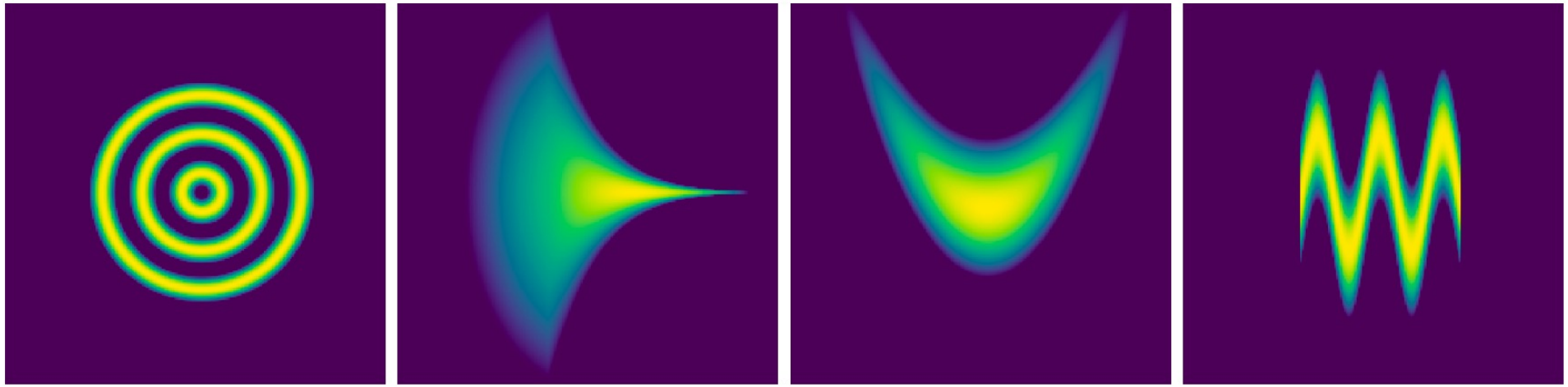
$\Rightarrow$ *increased expressiveness*

What Have we Learned from Logical Reasonings

Mixtures: recall OR (soft determinism)

Expressiveness: Ability to represent rich and effective classes of functions

$\Rightarrow$ *mixture of Gaussians can approximate any distribution!*

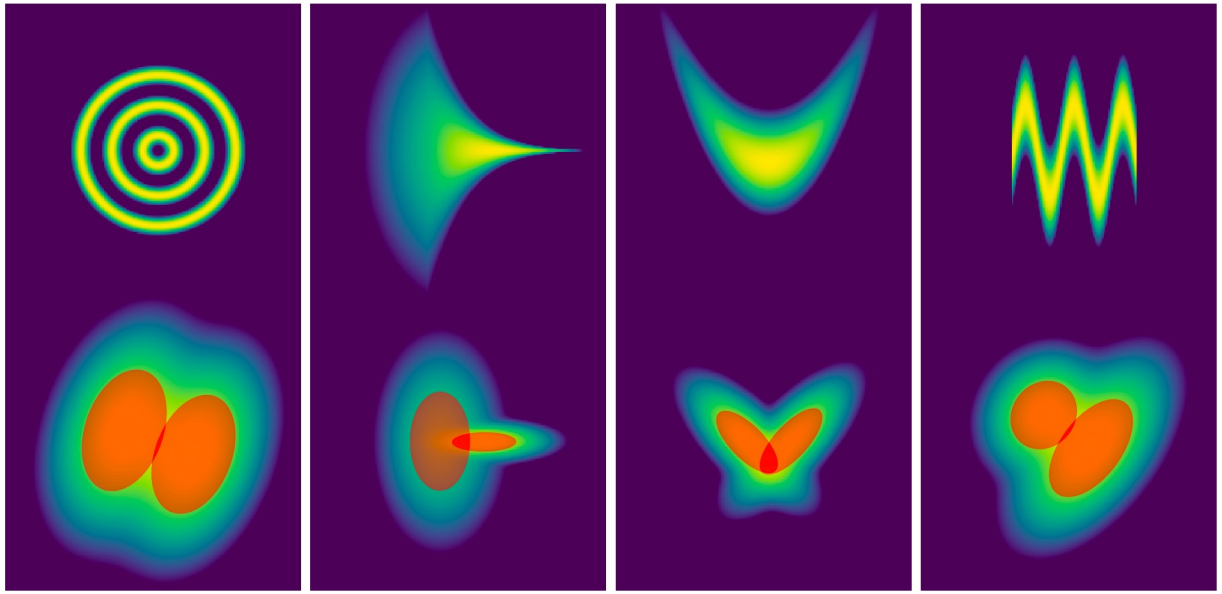


What Have we Learned from Logical Reasonings

Mixtures: recall OR (soft determinism)

Expressiveness: Ability to represent rich and effective classes of functions

$\Rightarrow$ *mixture of Gaussians can approximate any distribution!*

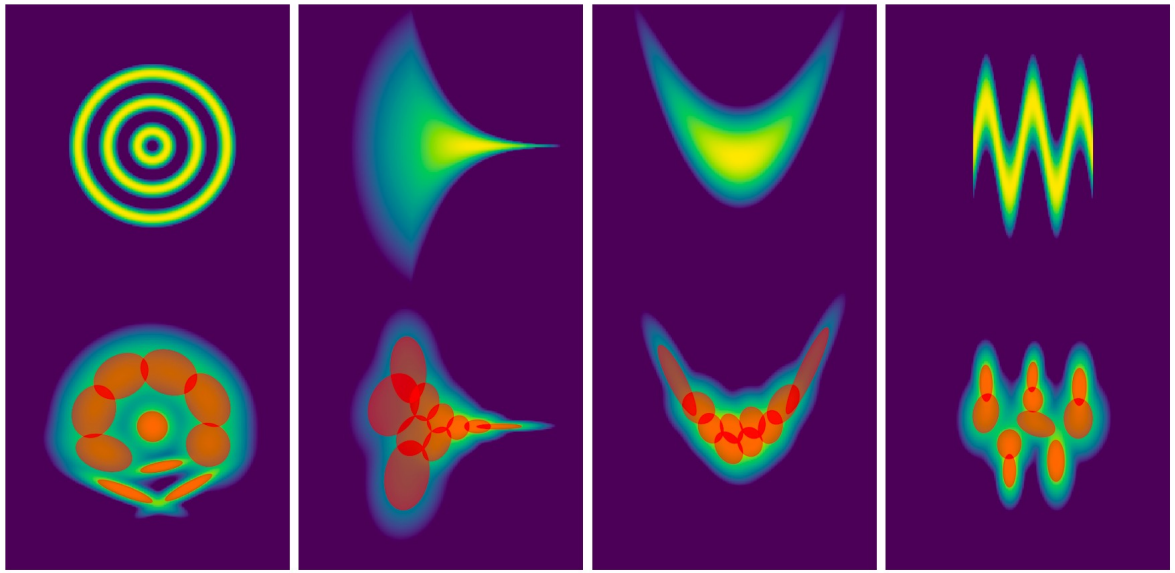


What Have we Learned from Logical Reasonings

Mixtures: recall OR (soft determinism)

Expressiveness: Ability to represent rich and effective classes of functions

$\Rightarrow$ *mixture of Gaussians can approximate any distribution!*

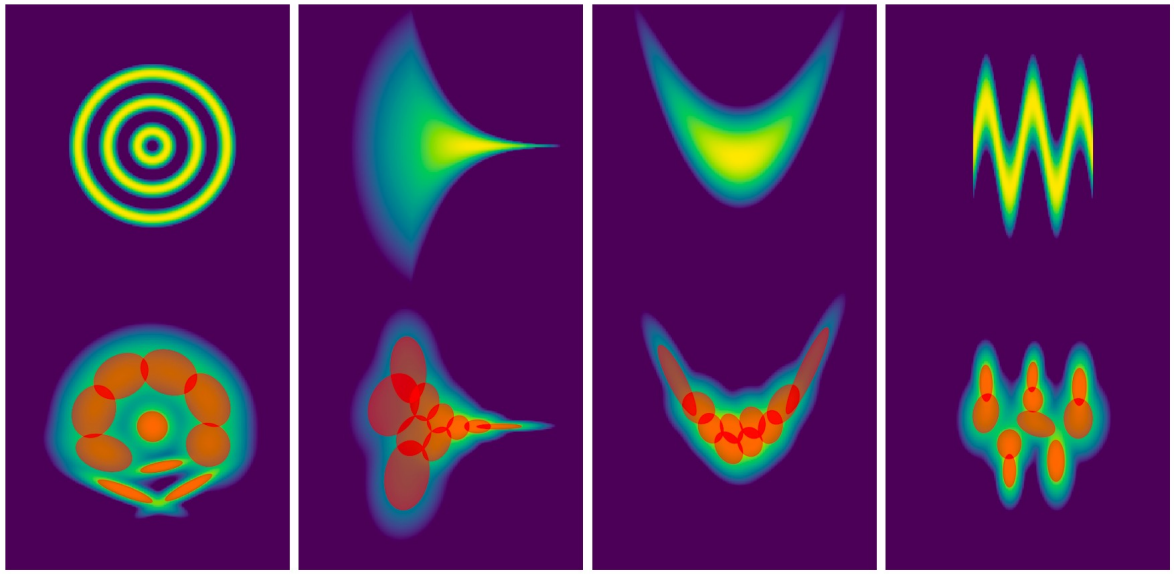


What Have we Learned from Logical Reasonings

Mixtures: recall OR (soft determinism)

Expressiveness: Ability to represent rich and effective classes of functions

$\Rightarrow$ *mixture of Gaussians can approximate any distribution!*



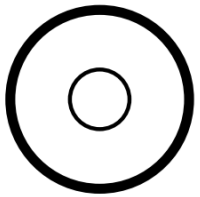
Tractable Reasoning Model: A Recipe

Base case: a single node encoding a distribution

$\Rightarrow$ *e.g., Gaussian PDF continuous random variable*

https://github.com/Juice-jl/

Tractable Reasoning Model: A Recipe



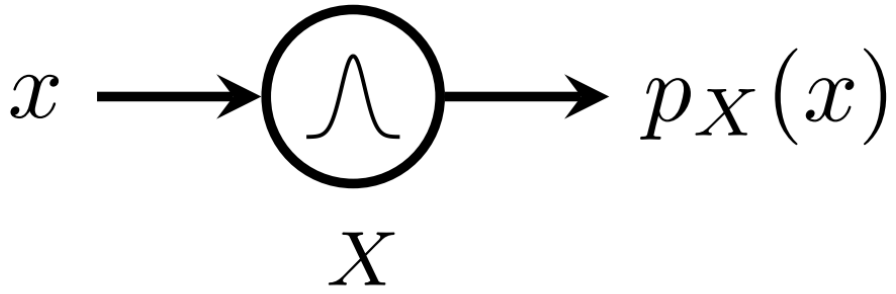
$\neg X$

Base case: a single node encoding a distribution

$\Rightarrow$ *e.g., indicators for X or $\neg X$ for Boolean random variable*

https://github.com/Juice-jl/

Tractable Reasoning Model: A Recipe



Simple distributions are tractable "black boxes" for:

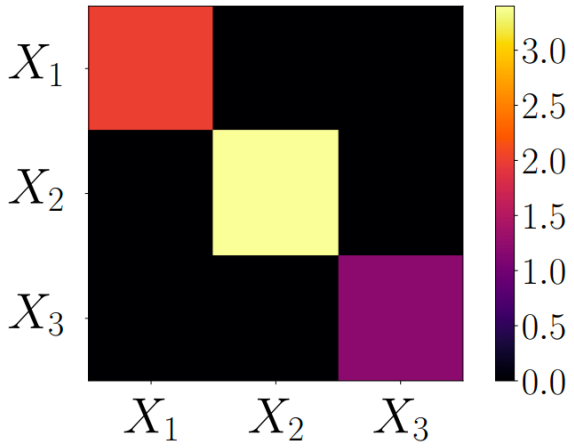
- EVI: output $p(\mathbf{x})$ (density or mass)
- MAR: output 1 (normalized) or Z (unnormalized)
- MAP: output the mode

https://github.com/Juice-jl/

Tractable Reasoning Model: A Recipe

Divide and conquer complexity

$$p(X_1, X_2, X_3) = p(X_1) \cdot p(X_2) \cdot p(X_3)$$

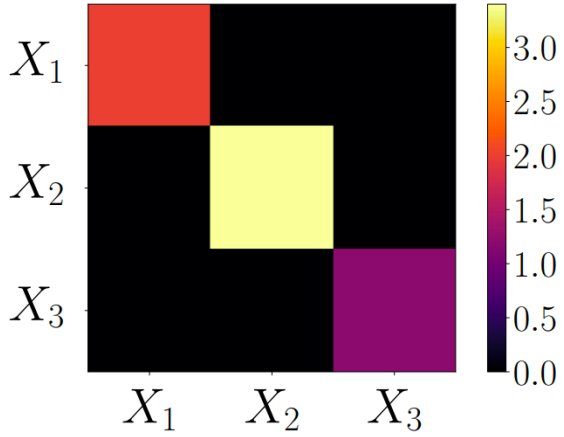


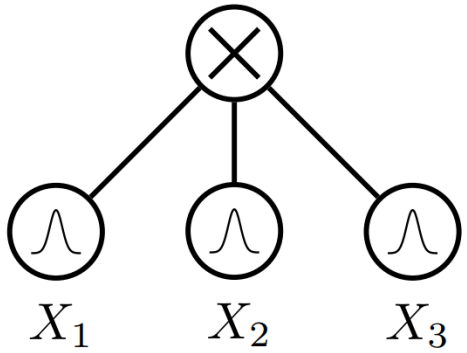
$\Rightarrow$ *e.g. modeling a multivariate Gaussian with diagonal covariance matrix…*

https://github.com/Juice-jl/

Tractable Reasoning Model: A Recipe

$$p(X_1, X_2, X_3) = p(X_1) \cdot p(X_2) \cdot p(X_3)$$



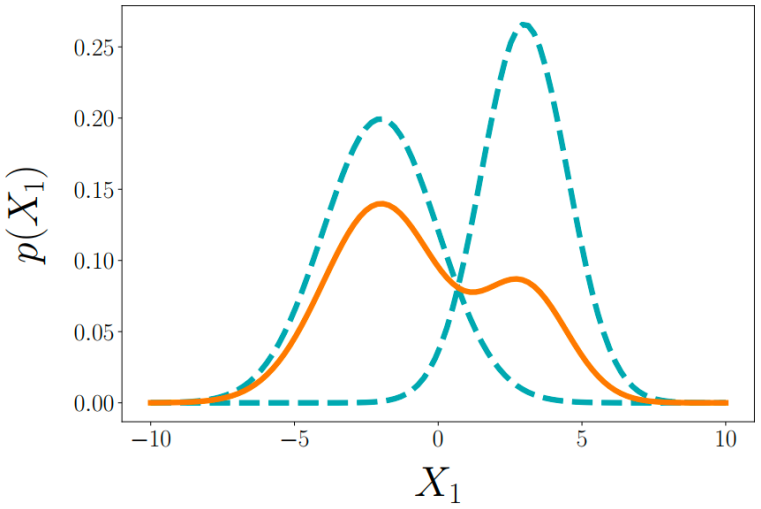


$\Rightarrow$ *...with a product node over some univariate Gaussian distribution*

https://github.com/Juice-jl/

Tractable Reasoning Model: A Recipe

Enhance expressiveness



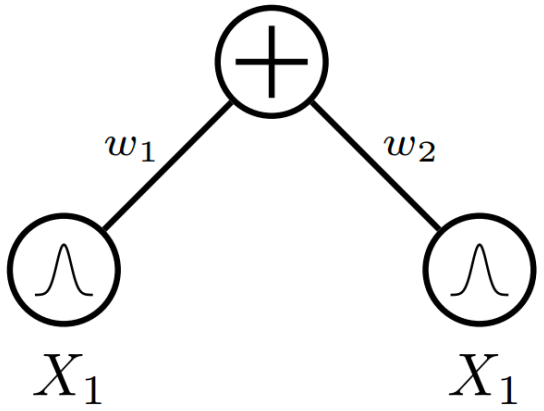
$$p(X) = w_1 \cdot p_1(X) + w_2 \cdot p_2(X)$$

⇒ *e.g. modeling a mixture of Gaussians…*

https://github.com/Juice-jl/

Tractable Reasoning Model: A Recipe

Enhance expressiveness



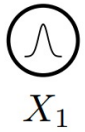
$$p(x) = 0.2 \cdot p_1(x) + 0.8 \cdot p_2(x)$$

$\Rightarrow$ *...as a weighted sum node over Gaussian input distributions*

https://github.com/Juice-jl/

Tractable Reasoning Model: A Recipe

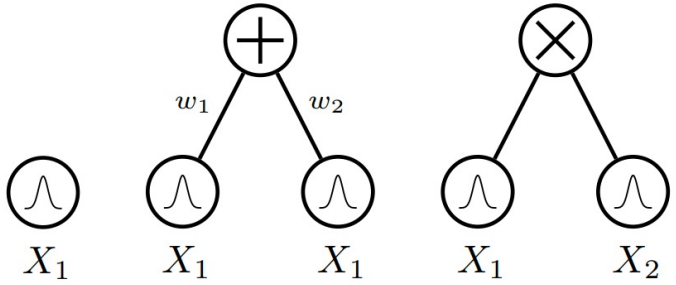
Recursive semantics of probabilistic circuits



https://github.com/Juice-jl/

Tractable Reasoning Model: A Recipe

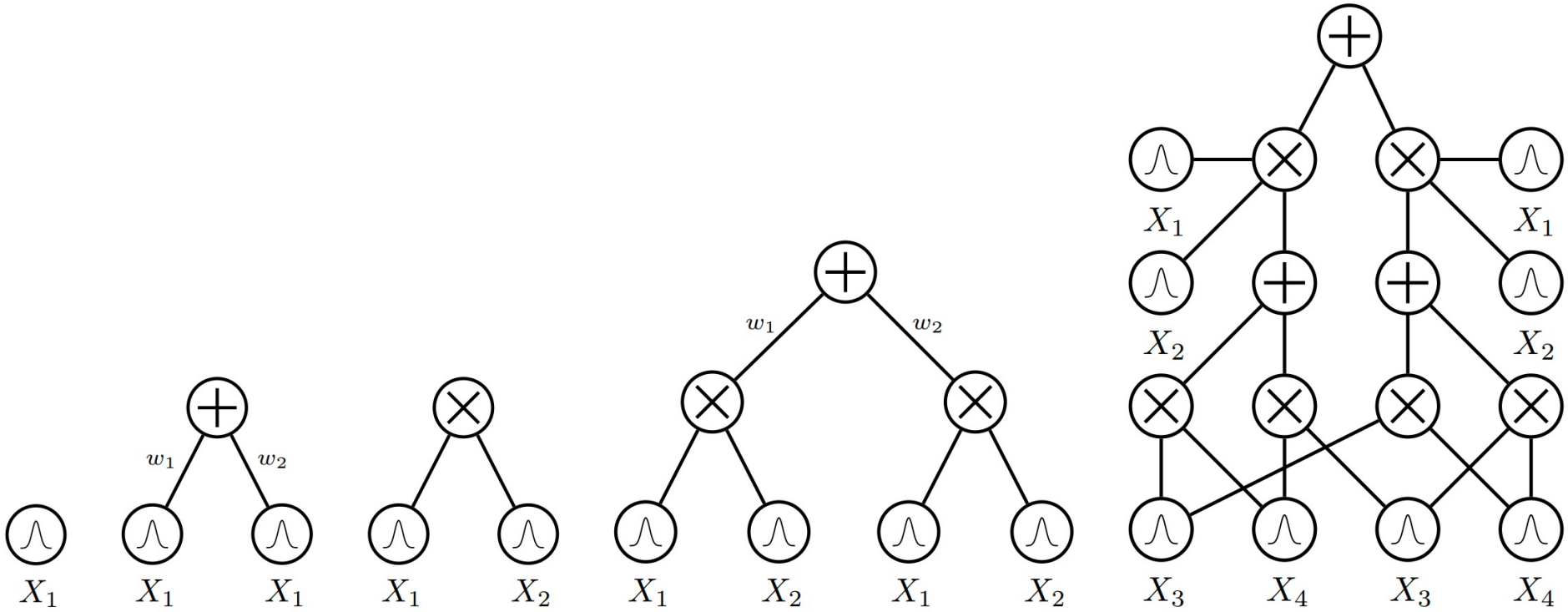
Recursive semantics of probabilistic circuits



https://github.com/Juice-jl/

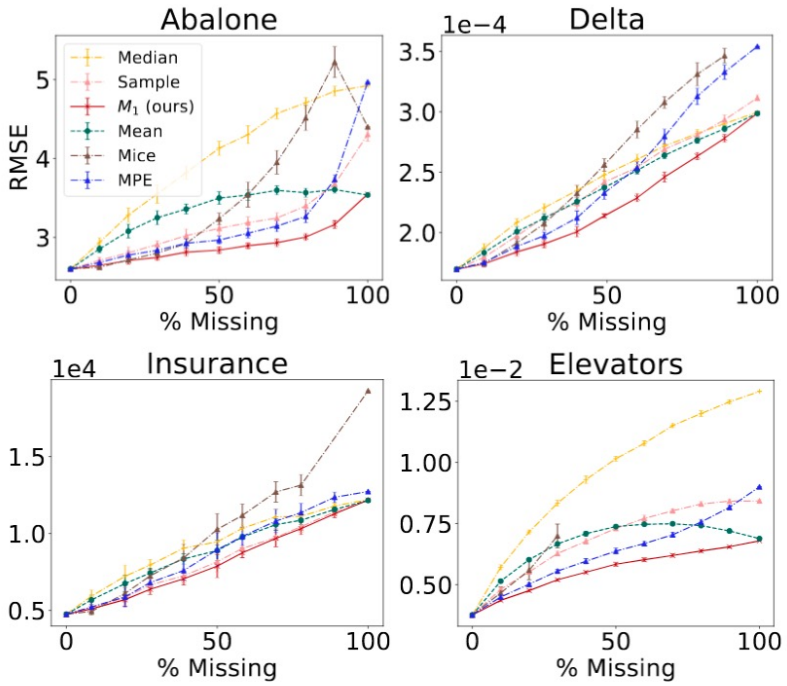
Tractable Reasoning Model: A Recipe

Recursive semantics of probabilistic circuits



https://github.com/Juice-jl/

An Example: Expected Predictions



Reasoning about the output of a classifier or regressor $\boldsymbol{f}$ given a distribution $\boldsymbol{p}$ over the input features

$$\mathop{\mathbb{E}}_{\mathbf{x}^m \sim p_\theta(\mathbf{x}^m \mid \mathbf{x}^o)} \left[f_\phi^k(\mathbf{x}^m, \mathbf{x}^o) \right]$$

$\Rightarrow$ **PR** *if* $\boldsymbol{f}$ *is a logical formula*

$\Rightarrow$ *missing values at test time*

$\Rightarrow$ *exploratory classifier analysis*

https://github.com/Juice-jl/

Khosravi et al., "On Tractable Computation of Expected Predictions", 2019

An Example: Expected Predictions

```
using ProbabilisticCircuits
pc = load_prob_circuit(zoo_psdd_file("insurance.psdd"));
rc = load_logistic_circuit(zoo_lc_file("insurance.circuit"), 1);
```

q_8: *How different is the insurance costs between smokers and non smokers?*

```
groups = make_observations([["!smoker"], ["smoker"]])
exps, _ = Expectation(pc, rc, groups);
println("Smoker     : \$ $(exps[2])");
println("Non-Smoker: \$ $(exps[1])");
println("Difference: \$ $(exps[2] - exps[1])");
Smoker     : $ 31355.32630488978
Non-Smoker: $ 8741.747258310648
Difference: $ 22613.57904657913
```

https://github.com/Juice-jl/

Khosravi et al., "On Tractable Computation of Expected Predictions", 2019

An Example: Expected Predictions

```
using ProbabilisticCircuits
pc = load_prob_circuit(zoo_psdd_file("insurance.psdd"));
rc = load_logistic_circuit(zoo_lc_file("insurance.circuit"), 1);
```

q_9: *Is the predictive model biased by gender?*

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

https://github.com/Juice-jl/

Khosravi et al., "On Tractable Computation of Expected Predictions", 2019

Conclusion: the AI Dilemma

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

Acknowledgements

Thanks to my collaborators!

Thanks for your attention!

Questions?