Mixing Logic and Deep Learning: The ‘Logic as Loss Function‘ Approach

Kyle Richardson

Allen Institute for AI

June 2022

The divide: logic vs. deep learning

Classical logic: designed for modeling *closed systems*.

The divide: logic vs. deep learning

Classical logic: designed for modeling *closed systems*.

$\forall x(0 \neq \mathbf{Succ}(x))$
$\forall x, y(\mathbf{Succ}(x) = \mathbf{Succ}(y)) \rightarrow x = y$
$\forall x(x + 0 = x)$
$\forall x, y(x + \mathbf{Succ}(y) = \mathbf{Succ}(x + y))$
$\forall x(x \cdot 0 = 0)$
$\forall \forall x, y(x \cdot \mathbf{Succ}(y) = x \cdot y + x)$

Theorem: $\forall x, y(x \cdot y = \mathbf{Succ}(0) \rightarrow x = \mathbf{Succ}(0) \wedge y = \mathbf{Succ}(0))$

The divide: logic vs. deep learning

Classical logic: designed for modeling *closed systems*.

$\forall x(0 \neq \mathbf{Succ}(x))$
$\forall x, y(\mathbf{Succ}(x) = \mathbf{Succ}(y)) \rightarrow x = y$
$\forall x(x + 0 = x)$
$\forall x, y(x + \mathbf{Succ}(y) = \mathbf{Succ}(x + y))$
$\forall x(x \cdot 0 = 0)$
$\forall \forall x, y(x \cdot \mathbf{Succ}(y) = x \cdot y + x)$

Theorem: $\forall x, y(x \cdot y = \mathbf{Succ}(0) \rightarrow x = \mathbf{Succ}(0) \wedge y = \mathbf{Succ}(0))$

deep learning: Used for modeling open systems, situations involving *partial information*, making (fast) predictions vs. explicit reasoning.

The divide: logic vs. deep learning

Classical logic: designed for modeling *closed systems*.

$$\forall x(0 \neq \mathbf{Succ}(x))$$
$$\forall x, y(\mathbf{Succ}(x) = \mathbf{Succ}(y)) \rightarrow x = y$$
$$\forall x(x + 0 = x)$$
$$\forall x, y(x + \mathbf{Succ}(y) = \mathbf{Succ}(x + y))$$
$$\forall x(x \cdot 0 = 0)$$
$$\forall \forall x, y(x \cdot \mathbf{Succ}(y) = x \cdot y + x)$$

Theorem: $\forall x, y(x \cdot y = \mathbf{Succ}(0) \rightarrow x = \mathbf{Succ}(0) \wedge y = \mathbf{Succ}(0))$

deep learning: Used for modeling open systems, situations involving *partial information*, making (fast) predictions vs. explicit reasoning.

Premise: A man with a hat is riding his bicycle down the street.
Hypothesis: A person is moving with the help of their legs.

Prediction: Entailment

Premise: A person is moving with the help of their legs.
Hypothesis: A person is moving

Prediction: Entailment

Two conceptual tools for relating logic and deep learning

1. Predictions-as-propositions

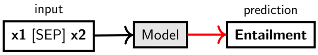
x1: A man with a hat is riding his bicycle down the street.
x2: A person is moving with the help of their legs.
Prediction: **Entailment**

x2: A person is moving with the help of their legs.
x3: A person is moving
Prediction: **Entailment**

1. Predictions-as-propositions

x1: A man with a hat is riding his bicycle down the street.
x2: A person is moving with the help of their legs.
Prediction: **Entailment**

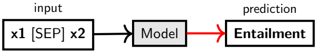
x2: A person is moving with the help of their legs.
x3: A person is moving
Prediction: **Entailment**

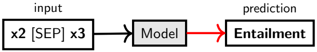


1. Predictions-as-propositions

x1: A man with a hat is riding his bicycle down the street.
x2: A person is moving with the help of their legs.
Prediction: **Entailment**

x2: A person is moving with the help of their legs.
x3: A person is moving
Prediction: **Entailment**



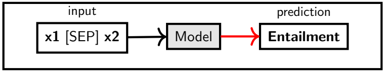


1. Predictions-as-propositions

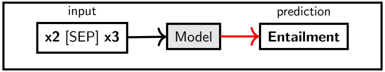
x1: A man with a hat is riding his bicycle down the street.
x2: A person is moving with the help of their legs.
Prediction: **Entailment**

x2: A person is moving with the help of their legs.
x3: A person is moving
Prediction: **Entailment**

symbolically: $\mathbf{E}(x_1, x_2)$



symbolically: $\mathbf{E}(x_2, x_3)$

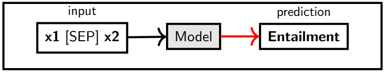


1. Predictions-as-propositions

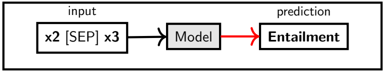
x1: A man with a hat is riding his bicycle down the street.
x2: A person is moving with the help of their legs.
Prediction: **Entailment**

x2: A person is moving with the help of their legs.
x3: A person is moving
Prediction: **Entailment**

symbolically: $\mathbf{E}(x_1, x_2)$



symbolically: $\mathbf{E}(x_2, x_3)$

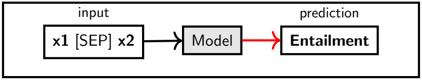


Proposition	Meaning
$\mathbf{E}(x, y)$	x *entails* y
$\mathbf{C}(x, y)$	x *contradicts* y

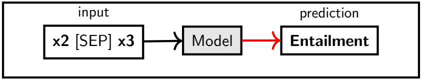
[Notation from Li et al. (2019)]

1. Predictions-as-propositions

symbolically: $\mathbf{E}(x_1, x_2)$



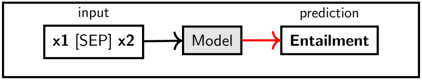
symbolically: $\mathbf{E}(x_2, x_3)$



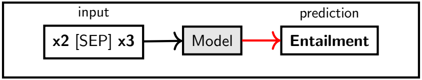
Why is this helpful?

1. Predictions-as-propositions

symbolically: $\mathbf{E}(x_1, x_2)$



symbolically: $\mathbf{E}(x_2, x_3)$



Why is this helpful?

constraints *(our expectations)*

$$\underbrace{\mathbf{E}(x_1, x_2) \land \mathbf{E}(x_2, x_3) \land \mathbf{C}(x_1, x_3)}_{\textbf{model predictions}} \longrightarrow \boxed{\forall x, y, z(\mathbf{E}(x, y) \land \mathbf{E}(y, z) \rightarrow \mathbf{E}(x, z))} \longrightarrow \text{satisfies constraints?}$$

Distinguishing good, bad and very bad model predictions

constraints *(our expectations)*
Minervini and Riedel (2018); Li et al. (2019)

$$\underbrace{\mathbf{E}(x_1, x_2) \wedge \mathbf{E}(x_2, x_3) \wedge \mathbf{C}(x_1, x_3)}_{\textbf{model predictions}} \longrightarrow \boxed{\begin{array}{c} \forall x, y, z(\mathbf{E}(x,y) \wedge \mathbf{E}(y,z) \rightarrow \mathbf{E}(x,z)) \\ \forall x, y, z(\mathbf{E}(x,y) \wedge \mathbf{C}(y,z) \rightarrow \mathbf{C}(x,z)) \\ \forall x, y(\mathbf{C}(x,y) \rightarrow \mathbf{C}(y,x)) \end{array}} \longrightarrow \text{satisfies constraints?}$$

Distinguishing good, bad and very bad model predictions

constraints *(our expectations)*
Minervini and Riedel (2018); Li et al. (2019)

$$\underbrace{\mathbf{E}(x_1, x_2) \wedge \mathbf{E}(x_2, x_3) \wedge \mathbf{C}(x_1, x_3)}_{\textbf{model predictions}} \longrightarrow \boxed{\begin{array}{c} \forall x, y, z(\mathbf{E}(x, y) \wedge \mathbf{E}(y, z) \rightarrow \mathbf{E}(x, z)) \\ \forall x, y, z(\mathbf{E}(x, y) \wedge \mathbf{C}(y, z) \rightarrow \mathbf{C}(x, z)) \\ \forall x, y(\mathbf{C}(x, y) \rightarrow \mathbf{C}(y, x)) \end{array}} \longrightarrow \text{satisfies constraints?}$$

The different predictions a model (or set of models) might make:

Distinguishing good, bad and very bad model predictions

constraints *(our expectations)*
Minervini and Riedel (2018); Li et al. (2019)

$$\underbrace{\mathbf{E}(x_1,x_2) \land \mathbf{E}(x_2,x_3) \land \mathbf{C}(x_1,x_3)}_{\textbf{model predictions}} \longrightarrow \boxed{\begin{array}{c} \forall x,y,z(\mathbf{E}(x,y) \land \mathbf{E}(y,z) \to \mathbf{E}(x,z)) \\ \forall x,y,z(\mathbf{E}(x,y) \land \mathbf{C}(y,z) \to \mathbf{C}(x,z)) \\ \forall x,y(\mathbf{C}(x,y) \to \mathbf{C}(y,x)) \end{array}} \longrightarrow \text{satisfies constraints?}$$

The different predictions a model (or set of models) might make:

x_1 : *A man with a hat is riding his bicycle down the street* **entails** x_2 : *A person is moving with the help of their legs*, x_3 : **entails** *A person is moving*

predictions	predictions correct	predictions consistent
$\mathbf{E}(x_1,x_2) \land \mathbf{E}(x_2,x_3) \land \mathbf{E}(x_1,x_3)$	✓ (3/3)	✓ (3/3)
$\mathbf{C}(x_1,x_2) \land \mathbf{C}(x_2,x_3) \land \mathbf{C}(x_1,x_3)$	✗ (0/3)	✓ (3/3)
$\mathbf{E}(x_1,x_2) \land \mathbf{C}(x_2,x_3) \land \mathbf{C}(x_1,x_3)$	✗ (1/3)	✓ (3/3)
$\mathbf{E}(x_1,x_2) \land \mathbf{E}(x_2,x_3) \land \mathbf{C}(x_1,x_3)$	✗ (2/3)	✗ (0/3)

Distinguishing good, bad and very bad model predictions

constraints *(our expectations)*
Minervini and Riedel (2018); Li et al. (2019)

$$\underbrace{\mathbf{E}(x_1, x_2) \wedge \mathbf{E}(x_2, x_3) \wedge \mathbf{C}(x_1, x_3)}_{\textbf{model predictions}} \longrightarrow \boxed{\begin{array}{c} \forall x, y, z(\mathbf{E}(x,y) \wedge \mathbf{E}(y,z) \rightarrow \mathbf{E}(x,z)) \\ \forall x, y, z(\mathbf{E}(x,y) \wedge \mathbf{C}(y,z) \rightarrow \mathbf{C}(x,z)) \\ \forall x, y(\mathbf{C}(x,y) \rightarrow \mathbf{C}(y,x)) \end{array}} \longrightarrow \text{satisfies constraints?}$$

The different predictions a model (or set of models) might make:

x_1 : *A man with a hat is riding his bicycle down the street* **entails** x_2 : *A person is moving with the help of their legs*, x_3 : **entails** *A person is moving*

predictions	predictions correct	predictions consistent
$\mathbf{E}(x_1, x_2) \wedge \mathbf{E}(x_2, x_3) \wedge \mathbf{E}(x_1, x_3)$	✓ (3/3)	✓ (3/3)
$\mathbf{C}(x_1, x_2) \wedge \mathbf{C}(x_2, x_3) \wedge \mathbf{C}(x_1, x_3)$	✗ (0/3)	✓ (3/3)
$\mathbf{E}(x_1, x_2) \wedge \mathbf{C}(x_2, x_3) \wedge \mathbf{C}(x_1, x_3)$	✗ (1/3)	✓ (3/3)
$\mathbf{E}(x_1, x_2) \wedge \mathbf{E}(x_2, x_3) \wedge \mathbf{C}(x_1, x_3)$	✗ (2/3)	✗ (0/3)

Thinking of predictions as **symbolic objects**; brings rigor to interpreting model behavior, can clarify what we mean by ‘good‘ vs. ‘bad‘.

Distinguishing good, bad and very bad model predictions

constraints *(our expectations)*
Minervini and Riedel (2018); Li et al. (2019)

$\underbrace{\mathbf{E}(x_1, x_2) \wedge \mathbf{E}(x_2, x_3) \wedge \mathbf{C}(x_1, x_3)}_{\textbf{model predictions}} \longrightarrow$

$$\boxed{\begin{array}{c} \forall x, y, z(\mathbf{E}(x,y) \wedge \mathbf{E}(y,z) \rightarrow \mathbf{E}(x,z)) \\ \forall x, y, z(\mathbf{E}(x,y) \wedge \mathbf{C}(y,z) \rightarrow \mathbf{C}(x,z)) \\ \forall x, y(\mathbf{C}(x,y) \rightarrow \mathbf{C}(y,x)) \end{array}}$$

$\longrightarrow$ satisfies constraints?

The different predictions a model (or set of models) might make:

x_1 : *A man with a hat is riding his bicycle down the street* **<u>entails</u>** x_2 : *A person is moving with the help of their legs*, x_3 : **<u>entails</u>** *A person is moving*

predictions	predictions correct	[redacted]
$\mathbf{E}(x_1, x_2) \wedge \mathbf{E}(x_2, x_3) \wedge \mathbf{E}(x_1, x_3)$	✓ (3/3)	
$\mathbf{C}(x_1, x_2) \wedge \mathbf{C}(x_2, x_3) \wedge \mathbf{C}(x_1, x_3)$	✗ (0/3)	
$\mathbf{E}(x_1, x_2) \wedge \mathbf{C}(x_2, x_3) \wedge \mathbf{C}(x_1, x_3)$	✗ (1/3)	
$\mathbf{E}(x_1, x_2) \wedge \mathbf{E}(x_2, x_3) \wedge \mathbf{C}(x_1, x_3)$	✗ (2/3)	

Thinking of predictions as **symbolic objects**; brings rigor to interpreting model behavior, can clarify what we mean by ‘good‘ vs. ‘bad‘.

Annotations-as-logical-specifications: what we expect

Question: (q) Where is a frisbee in play likely to be? (m) 1) **air** 2) ...

Prediction $(q, m \rightarrow a)$: (a) *"air"*
Prediction $(q, m \rightarrow e, a)$: (e) *"A frisbee floats on air"*, *"air"*
Prediction $(q, m \rightarrow p)$: (p) *"A frisby in play is likely to be in the air"*

Annotations-as-logical-specifications: what we expect

Question: (q) Where is a frisbee in play likely to be? (m) 1) **air** 2) ...

Prediction $(q, m \rightarrow a)$**:** (a) *"air"*
Prediction $(q, m \rightarrow e, a)$**:** (e) *"A frisbee floats on air"*, *"air"*
Prediction $(q, m \rightarrow p)$**:** (p) *"A frisby in play is likely to be in the air"*

(Labeled) datasets specify what we want a model to do and learn.

instance	**meaning**	**proposition** (logic)
(q, m, a)	*a is the correct answer to q in m*	$\mathbf{Q}(q, a)$
(q, m, e, a)	*e is the correct explanation of q with answer a*	$\mathbf{Ex}(q, e + a)$
(q, m, p)	*p is the correct proposition corresponding to q and a.*	$\mathbf{P}(q, p)$

Annotations-as-logical-specifications: what we expect

Question: (q) Where is a frisbee in play likely to be? (m) 1) **air** 2) ...

Prediction $(q, m \rightarrow a)$**:** (a) *"air"*
Prediction $(q, m \rightarrow e, a)$**:** (e) *"A frisbee floats on air", "air"*
Prediction $(q, m \rightarrow p)$**:** (p) *"A frisby in play is likely to be in the air"*

(Labeled) datasets specify what we want a model to do and learn.

instance	**meaning**	**proposition** (logic)
(q, m, a)	*a is the correct answer to q in m*	$\mathbf{Q}(q, a)$
(q, m, e, a)	*e is the correct explanation of q with answer a*	$\mathbf{Ex}(q, e + a)$
(q, m, p)	*p is the correct proposition corresponding to q and a.*	$\mathbf{P}(q, p)$

We can also think of annotations as logical propositions and formulae.

$$\mathbf{Q}(q, a) \wedge \mathbf{Ex}(q, e + a) \wedge \mathbf{P}(q, p)$$

Annotations-as-logic-specifications: what we expect

Question: (q) Where is a frisbee in play likely to be? (m) 1) **air** 2) ...

Prediction ($q, m \rightarrow a$): (a) *"air"*
Prediction ($q, m \rightarrow e, a$): (e) *"A frisbee floats on air", "air"*
Prediction ($q, m \rightarrow p$): (p) *"A frisby in play is likely to be in the air"*

$$\underbrace{\mathbf{Q}(q,a) \land \mathbf{Ex}(q, e+a) \land \mathbf{P}(q,p)}_{\text{should make correct predictions}} \land \boxed{\underbrace{\forall q,p(\mathbf{P}(q,p) \rightarrow \mathbf{Bel}(p))}_{\text{should believe propositions}} \land \underbrace{\forall q,a,e(\mathbf{Ex}(q,e+a) \rightarrow \mathbf{Q}(q+e,a))}_{\text{Answer should be invariant given its explanation}}}$$

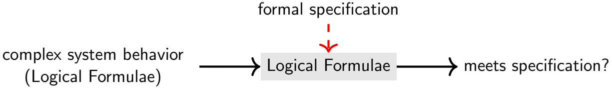
Why is this helpful? Be more clear about what we expect, understand gap between what we expect and what we *actually* do, imagine new tasks.

Conceptual Tools: Predictions-as-propositions, Annotations-as-specifications

Thinking of *model predictions as symbolic objects*; *annotations and our expectations* as logical formulas.

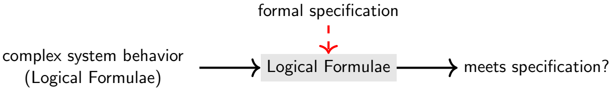
Conceptual Tools: Predictions-as-propositions, Annotations-as-specifications

Thinking of *model predictions as symbolic objects*; *annotations and our expectations* as logical formulas.



Conceptual Tools: Predictions-as-propositions, Annotations-as-specifications

Thinking of *model predictions as symbolic objects*; *annotations and our expectations* as logical formulas.



note: These are just conceptual tools, not particularly useful yet.

Technical Tool: Multi-valued Logic and the 'Logic as Loss Function' approach (see review in Marra et al. (2021)).

Multi-valued logic

Logical propositions come in different flavors

Boolean Propositions	$\mathbf{P} \in \{0, 1\}$
(Probabilistic) Boolean Propositions	$\mathbf{P}$ is 1 with prob. $p \in [0, 1]$ (0 with prob. $1 - p$)
non-classical logic	
(Finite-)N-Valued Propositions	$\mathbf{P} \in \{0, 1, \ldots N\}$
Real-Valued (Fuzzy) Propositions	$\mathbf{P} \in [0, 1]$ (*truth degree*),

Multi-valued logic

Logical propositions come in different flavors

Boolean Propositions	$\mathbf{P} \in \{0, 1\}$
(Probabilistic) Boolean Propositions	$\mathbf{P}$ is 1 with prob. $p \in [0, 1]$ (0 with prob. $1 - p$)
non-classical logic	
(Finite-)N-Valued Propositions	$\mathbf{P} \in \{0, 1, \ldots N\}$
Real-Valued (Fuzzy) Propositions	$\mathbf{P} \in [0, 1]$ (*truth degree*),

Fuzzy Logic: Logical operators $(\wedge, \vee, \neg, \rightarrow)$ are turned into real-valued functions (t-norms); generalizes Boolean logic to continuous values.

Multi-valued logic

Logical propositions come in different flavors

Boolean Propositions	$\mathbf{P} \in \{0,1\}$
(Probabilistic) Boolean Propositions	$\mathbf{P}$ is 1 with prob. $p \in [0,1]$ (0 with prob. $1-p$)
non-classical logic	
(Finite-)N-Valued Propositions	$\mathbf{P} \in \{0,1,\ldots N\}$
Real-Valued (Fuzzy) Propositions	$\mathbf{P} \in [0,1]$ (*truth degree*),

Fuzzy Logic: Logical operators $(\wedge, \vee, \neg, \rightarrow)$ are turned into real-valued functions (t-norms); generalizes Boolean logic to continuous values.

see Li et al. (2019); Grespan et al. (2021)

	Boolean Logic	Product	Łukasiewisz	Gödel
T-norm	$\mathbf{P_1} \wedge \mathbf{P}_2$	$\mathbf{P}_1 \cdot \mathbf{P}_2$	$\max(0, \mathbf{P}_1 + \mathbf{P}_2 - 1)$	$\min(\mathbf{P}_1, \mathbf{P}_2)$
T-conorm	$\mathbf{P_1} \vee \mathbf{P}_2$	$\mathbf{P}_1 + \mathbf{P}_2 - \mathbf{P}_1 \cdot \mathbf{P}_2$	$\min(1, \mathbf{P}_1, +\mathbf{P}_2)$	$\max(\mathbf{P}_1, \mathbf{P}_2)$
Negation	$\neg\mathbf{P}$	$1 - \mathbf{P}$	$1 - \mathbf{P}$	$1 - \mathbf{P}$
Residuum	$\mathbf{P}_1 \rightarrow \mathbf{P}_2$	$\min(1, \frac{\mathbf{P}_2}{\mathbf{P}_1})$	$\min(1, 1 - \mathbf{P}_1 + \mathbf{P}_2)$	$\mathbf{P}_2$

Multi-valued logic

Logical propositions come in different flavors

Boolean Propositions	$\mathbf{P} \in \{0, 1\}$
(Probabilistic) Boolean Propositions	$\mathbf{P}$ is 1 with prob. $p \in [0, 1]$ (0 with prob. $1 - p$)
non-classical logic	
(Finite-)N-Valued Propositions	$\mathbf{P} \in \{0, 1, \ldots N\}$
Real-Valued (Fuzzy) Propositions	$\mathbf{P} \in [0, 1]$ (*truth degree*),

Fuzzy Logic: Logical operators ($\wedge, \vee, \neg, \rightarrow$) are turned into real-valued functions (t-norms); generalizes Boolean logic to continuous values.

see Li et al. (2019); Grespan et al. (2021)

	Boolean Logic	Product	Łukasiewisz	Gödel
T-norm	$\mathbf{P_1} \wedge \mathbf{P}_2$	$\mathbf{P}_1 \cdot \mathbf{P}_2$	$\max(0, \mathbf{P}_1 + \mathbf{P}_2 - 1)$	$\min(\mathbf{P}_1, \mathbf{P}_2)$
T-conorm	$\mathbf{P_1} \vee \mathbf{P}_2$	$\mathbf{P}_1 + \mathbf{P}_2 - \mathbf{P}_1 \cdot \mathbf{P}_2$	$\min(1, \mathbf{P}_1, +\mathbf{P}_2)$	$\max(\mathbf{P}_1, \mathbf{P}_2)$
Negation	$\neg\mathbf{P}$	$1 - \mathbf{P}$	$1 - \mathbf{P}$	$1 - \mathbf{P}$
Residuum	$\mathbf{P}_1 \rightarrow \mathbf{P}_2$	$\min(1, \frac{\mathbf{P}_2}{\mathbf{P}_1})$	$\min(1, 1 - \mathbf{P}_1 + \mathbf{P}_2)$	$\mathbf{P}_2$

Note: *At the extremes* 0 *and* 1, *work exactly as classical counterparts.*

Turning Specifications into Loss Functions

	Boolean Logic	Product	Łukasiewisz	Gödel
T-norm	$\mathbf{P_1} \wedge \mathbf{P}_2$	$\mathbf{P}_1 \cdot \mathbf{P}_2$	$\max(0, \mathbf{P}_1 + \mathbf{P}_2 - 1)$	$\min(\mathbf{P}_1, \mathbf{P}_2)$
T-conorm	$\mathbf{P_1} \vee \mathbf{P}_2$	$\mathbf{P}_1 + \mathbf{P}_2 - \mathbf{P}_1 \cdot \mathbf{P}_2$	$\min(1, \mathbf{P}_1, +\mathbf{P}_2)$	$\max(\mathbf{P}_1, \mathbf{P}_2)$
Negation	$\neg\mathbf{P}$	$1 - \mathbf{P}$	$1 - \mathbf{P}$	$1 - \mathbf{P}$
Residuum	$\mathbf{P}_1 \rightarrow \mathbf{P}_2$	$\min(1, \frac{\mathbf{P}_2}{\mathbf{P}_1})$	$\min(1, 1 - \mathbf{P}_1 + \mathbf{P}_2)$	$\mathbf{P}_2$

Why are these (T-norms) useful?

Turning Specifications into Loss Functions

	Boolean Logic	Product	Łukasiewisz	Gödel
T-norm	$\mathbf{P_1} \wedge \mathbf{P_2}$	$\mathbf{P}_1 \cdot \mathbf{P}_2$	$\max(0, \mathbf{P}_1 + \mathbf{P}_2 - 1)$	$\min(\mathbf{P}_1, \mathbf{P}_2)$
T-conorm	$\mathbf{P_1} \vee \mathbf{P_2}$	$\mathbf{P}_1 + \mathbf{P}_2 - \mathbf{P}_1 \cdot \mathbf{P}_2$	$\min(1, \mathbf{P}_1, +\mathbf{P}_2)$	$\max(\mathbf{P}_1, \mathbf{P}_2)$
Negation	$\neg\mathbf{P}$	$1 - \mathbf{P}$	$1 - \mathbf{P}$	$1 - \mathbf{P}$
Residuum	$\mathbf{P}_1 \rightarrow \mathbf{P}_2$	$\min(1, \frac{\mathbf{P}_2}{\mathbf{P}_1})$	$\min(1, 1 - \mathbf{P}_1 + \mathbf{P}_2)$	$\mathbf{P}_2$

Why are these (T-norms) useful?

Specifications

$$\underbrace{\mathbf{Q}(q,a) \wedge \mathbf{Ex}(q,e) \wedge \mathbf{P}(q,p)}_{\text{should make correct predictions}} \wedge \underbrace{(\mathbf{P}(q,p) \rightarrow \mathbf{Bel}(p))}_{\text{should believe propositions}} \wedge \underbrace{(\mathbf{Ex}(q,e+a) \rightarrow \mathbf{Q}(q+e,a))}_{\text{Answer should be invariant given its explanation}}$$

$\downarrow$ *(translation to T-norms)*

Differentiable Loss Function

Turning Specifications into Loss Functions

	Boolean Logic	Product	Łukasiewisz	Gödel
T-norm	$\mathbf{P_1} \land \mathbf{P_2}$	$\mathbf{P}_1 \cdot \mathbf{P}_2$	$\max(0, \mathbf{P}_1 + \mathbf{P}_2 - 1)$	$\min(\mathbf{P}_1, \mathbf{P}_2)$
T-conorm	$\mathbf{P_1} \lor \mathbf{P}_2$	$\mathbf{P}_1 + \mathbf{P}_2 - \mathbf{P}_1 \cdot \mathbf{P}_2$	$\min(1, \mathbf{P}_1, +\mathbf{P}_2)$	$\max(\mathbf{P}_1, \mathbf{P}_2)$
Negation	$\neg\mathbf{P}$	$1 - \mathbf{P}$	$1 - \mathbf{P}$	$1 - \mathbf{P}$
Residuum	$\mathbf{P}_1 \rightarrow \mathbf{P}_2$	$\min(1, \frac{\mathbf{P}_2}{\mathbf{P}_1})$	$\min(1, 1 - \mathbf{P}_1 + \mathbf{P}_2)$	$\mathbf{P}_2$

Why are these (T-norms) useful?

Specifications

$$\underbrace{\mathbf{Q}(q,a) \land \mathbf{Ex}(q,e) \land \mathbf{P}(q,p)}_{\text{should make correct predictions}} \land \underbrace{(\mathbf{P}(q,p) \rightarrow \mathbf{Bel}(p))}_{\text{should believe propositions}} \land \underbrace{(\mathbf{Ex}(q, e+a) \rightarrow \mathbf{Q}(q+e, a))}_{\text{Answer should be invariant given its explanation}}$$

(translation to T-norms)

Differentiable Loss Function

$$\mathbf{Loss}_{model}\Big(\mathbf{Q}(q,a) \land \mathbf{Ex}(q,e) \land \ldots\Big) = \underbrace{\sum_{\mathbf{P} \in \{\mathbf{Q}(q,a), \mathbf{Ex}(q,e), \ldots\}} - \log \underbrace{p_{model}(\mathbf{P})}_{\text{fuzzy truth degree}}}_{\textbf{Product T-norm}\ (-\log\text{ prob.})}$$

Training Objectives as Logical Specifications

$$\underbrace{\mathbf{Q}(q,a) \wedge \mathbf{Ex}(q,e) \wedge \mathbf{P}(q,p)}_{\text{Atomic predictions}} \wedge \underbrace{\underbrace{(\mathbf{P}(q,p) \rightarrow \mathbf{Bel}(p))}_{\text{should believe propositions}} \wedge \underbrace{(\mathbf{Ex}(q,e+a) \rightarrow \mathbf{Q}(q+e,a))}_{\text{Answer should be invariant given its explanation}}}_{\text{Additional Constaints}}$$

We can think of a (supervised) dataset $D = \{(x_j, y_j)\}_{j=0}^{N}$ as a set of true atomic propositions propositions $D_p = \{\mathbf{Y}_1, \ldots, \mathbf{Y}_N\}$ with constraints C.

Goal	Logical Formula	Loss Function *(Product)*
Make Correct Predictions	$\bigwedge_{\mathbf{Y} \in D_p} \mathbf{Y}$	$\sum_{\mathbf{Y} \in D_p} -\log p_{model}(\mathbf{Y})$
Believe your propositions	$\bigwedge_{\mathbf{P}(q,p) \in D_p} \mathbf{P}(q,p) \rightarrow \mathbf{Bel}(p)$	$\sum_{\mathbf{P}(q,p) \in D_p} \mathbf{ReLU}(\log p_{model}(\mathbf{P}(q,p)) - \log p_{model}(\mathbf{Bel}(p)))$

Training Objectives as Logical Specifications

$$\underbrace{\mathbf{Q}(q,a) \wedge \mathbf{Ex}(q,e) \wedge \mathbf{P}(q,p)}_{\text{Atomic predictions}} \wedge \underbrace{\underbrace{(\mathbf{P}(q,p) \rightarrow \mathbf{Bel}(p))}_{\text{should believe propositions}} \wedge \underbrace{(\mathbf{Ex}(q,e+a) \rightarrow \mathbf{Q}(q+e,a))}_{\text{Answer should be invariant given its explanation}}}_{\text{Additional Constaints}}$$

We can think of a (supervised) dataset $D = \{(x_j, y_j)\}_{j=0}^{N}$ as a set of true atomic propositions propositions $D_p = \{\mathbf{Y}_1, ..., \mathbf{Y}_N\}$ with constraints C.

Goal	Logical Formula	Loss Function *(Product)*
Make Correct Predictions	$\bigwedge_{\mathbf{Y} \in D_p} \mathbf{Y}$	$\sum_{\mathbf{Y} \in D_p} -\log p_{model}(\mathbf{Y})$
Believe your propositions	$\bigwedge_{\mathbf{P}(q,p) \in D_p} \mathbf{P}(q,p) \rightarrow \mathbf{Bel}(p)$	$\sum_{\mathbf{P}(q,p) \in D_p} \mathbf{ReLU}(\log p_{model}(\mathbf{P}(q,p)) - \log p_{model}(\mathbf{Bel}(p)))$

Observation (Rocktäschel et al., 2015; Li et al., 2019): Translating product conjunction to negative log space yields ordinary **cross-entropy** loss.

Training Objectives as Logical Specifications

$$\underbrace{\mathbf{Q}(q,a) \wedge \mathbf{Ex}(q,e) \wedge \mathbf{P}(q,p)}_{\text{Atomic predictions}} \wedge \underbrace{\underbrace{(\mathbf{P}(q,p) \rightarrow \mathbf{Bel}(p))}_{\text{should believe propositions}} \wedge \underbrace{(\mathbf{Ex}(q,e+a) \rightarrow \mathbf{Q}(q+e,a))}_{\text{Answer should be invariant given its explanation}}}_{\text{Additional Constaints}}$$

We can think of a (supervised) dataset $D = \{(x_j, y_j)\}_{j=0}^{N}$ as a set of true atomic propositions propositions $D_p = \{\mathbf{Y}_1, ..., \mathbf{Y}_N\}$ with constraints C.

Goal	Logical Formula	Loss Function *(Product)*
Make Correct Predictions	$\bigwedge_{\mathbf{Y} \in D_p} \mathbf{Y}$	$\sum_{\mathbf{Y} \in D_p} -\log p_{model}(\mathbf{Y})$
Believe your propositions	$\bigwedge_{\mathbf{P}(q,p) \in D_p} \mathbf{P}(q,p) \rightarrow \mathbf{Bel}(p)$	$\sum_{\mathbf{P}(q,p) \in D_p} \mathbf{ReLU}(\log p_{model}(\mathbf{P}(q,p)) - \log p_{model}(\mathbf{Bel}(p)))$

Observation: There is often a large gap between what we train models to do (e.g., **make correct predictions**) and we expect models to know.

Logic as Loss Function, Logic in the Weights

Logical constraints serve as regularizer, *soft constraints* over hypothesis space that favor solutions closer to knowledge (undirected models).

$$\textbf{Loss} = \underbrace{\textbf{Loss}_{\mathrm{CE}}}_{\text{ordinary loss}} + \underbrace{\lambda \textbf{Loss}_{\textit{constraints}}}_{\text{logical constraints}}$$

Logic as Loss Function, Logic in the Weights

Logical constraints serve as regularizer, *soft constraints* over hypothesis space that favor solutions closer to knowledge (undirected models).

$$\textbf{Loss} = \underbrace{\textbf{Loss}_{\mathrm{CE}}}_{\text{ordinary loss}} + \underbrace{\lambda \textbf{Loss}_{\textit{constraints}}}_{\text{logical constraints}}$$

$$\underbrace{\mathbf{E}(x_1, x_2) \land \mathbf{E}(x_2, x_3) \land \mathbf{C}(x_1, x_3)}_{\textbf{model predictions}} \longrightarrow \boxed{\begin{array}{c} \forall x, y, z(\mathbf{E}(x,y) \land \mathbf{E}(y,z) \to \mathbf{E}(x,z)) \\ \forall x, y, z(\mathbf{E}(x,y) \land \mathbf{C}(y,z) \to \mathbf{C}(x,z)) \\ \forall x, y(\mathbf{C}(x,y) \to \mathbf{C}(y,x)) \end{array}} \xrightarrow{\text{t-norms}} \textbf{Loss}_{\textit{constraints}}$$

predictions	predictions correct	predictions consistent	prediction loss	constraint loss
$\mathbf{E}(x_1, x_2) \land \mathbf{E}(x_2, x_3) \land \mathbf{E}(x_1, x_3)$	✓ (3/3)	✓ (3/3)	low	low
$\mathbf{C}(x_1, x_2) \land \mathbf{C}(x_2, x_3) \land \mathbf{C}(x_1, x_3)$	✗ (0/3)	✓ (3/3)	high	low
$\mathbf{E}(x_1, x_2) \land \mathbf{C}(x_2, x_3) \land \mathbf{C}(x_1, x_3)$	✗ (1/3)	✓ (3/3)	high	low
$\mathbf{E}(x_1, x_2) \land \mathbf{E}(x_2, x_3) \land \mathbf{C}(x_1, x_3)$	✗ (2/3)	✗ (0/3)	medium	high

Logic as Loss Function, Logic in the Weights

Logical constraints serve as regularizer, *soft constraints* over hypothesis space that favor solutions closer to knowledge (undirected models).

$$\textbf{Loss} = \underbrace{\textbf{Loss}_{\mathrm{CE}}}_{\text{ordinary loss}} + \underbrace{\lambda\textbf{Loss}_{constraints}}_{\text{logical constraints}}$$

$$\underbrace{\mathbf{E}(x_1,x_2) \wedge \mathbf{E}(x_2,x_3) \wedge \mathbf{C}(x_1,x_3)}_{\textbf{model predictions}} \longrightarrow \boxed{\begin{array}{c} \forall x,y,z(\mathbf{E}(x,y) \wedge \mathbf{E}(y,z) \rightarrow \mathbf{E}(x,z)) \\ \forall x,y,z(\mathbf{E}(x,y) \wedge \mathbf{C}(y,z) \rightarrow \mathbf{C}(x,z)) \\ \forall x,y(\mathbf{C}(x,y) \rightarrow \mathbf{C}(y,x)) \end{array}} \xrightarrow{\text{t-norms}} \textbf{Loss}_{constraints}$$

predictions	**predictions correct**	**predictions consistent**	**prediction loss**	**constraint loss**
$\mathbf{E}(x_1,x_2) \wedge \mathbf{E}(x_2,x_3) \wedge \mathbf{E}(x_1,x_3)$	✓ (3/3)	✓ (3/3)	low	low
$\mathbf{C}(x_1,x_2) \wedge \mathbf{C}(x_2,x_3) \wedge \mathbf{C}(x_1,x_3)$	✗ (0/3)	✓ (3/3)	high	low
$\mathbf{E}(x_1,x_2) \wedge \mathbf{C}(x_2,x_3) \wedge \mathbf{C}(x_1,x_3)$	✗ (1/3)	✓ (3/3)	high	low
$\mathbf{E}(x_1,x_2) \wedge \mathbf{E}(x_2,x_3) \wedge \mathbf{C}(x_1,x_3)$	✗ (2/3)	✗ (0/3)	medium	high

Observation: constraint loss doesn't contribute much outside of penalizing the last case; need to be carefully constructed.

Logic as Loss Function, Logic in the Weights

Logical constraints serve as regularizer, *soft constraints* over hypothesis space that favor solutions closer to knowledge (undirected models).

$$\textbf{Loss} = \underbrace{\textbf{Loss}_{\mathrm{CE}}}_{\text{ordinary loss}} + \underbrace{\lambda\textbf{Loss}_{\textit{constraints}}}_{\text{logical constraints}}$$

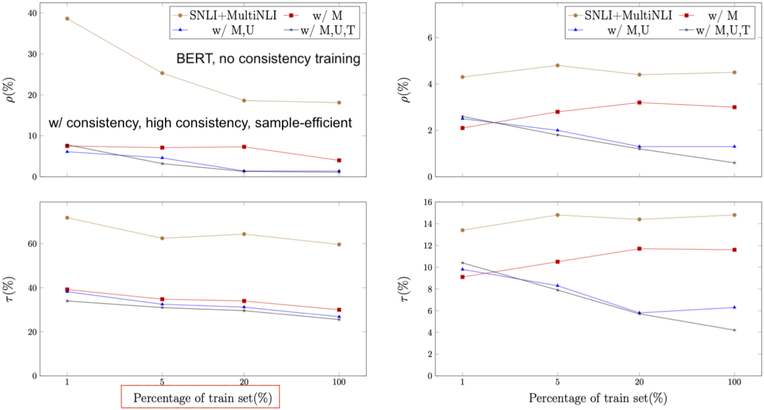
$$\underbrace{\mathbf{E}(x_1,x_2) \wedge \mathbf{E}(x_2,x_3) \wedge \mathbf{C}(x_1,x_3)}_{\textbf{model predictions}} \longrightarrow \boxed{\begin{array}{c} \forall x,y,z(\mathbf{E}(x,y) \wedge \mathbf{E}(y,z) \rightarrow \mathbf{E}(x,z)) \\ \forall x,y,z(\mathbf{E}(x,y) \wedge \mathbf{C}(y,z) \rightarrow \mathbf{C}(x,z)) \\ \forall x,y(\mathbf{C}(x,y) \rightarrow \mathbf{C}(y,x)) \end{array}} \xrightarrow{\text{t-norms}} \textbf{Loss}_{\textit{constraints}}$$

predictions	[redacted]	**predictions consistent**	[redacted]	**constraint loss**
$\mathbf{E}(x_1,x_2) \wedge \mathbf{E}(x_2,x_3) \wedge \mathbf{E}(x_1,x_3)$	[redacted]	✓ (3/3)	[redacted]	low
$\mathbf{C}(x_1,x_2) \wedge \mathbf{C}(x_2,x_3) \wedge \mathbf{C}(x_1,x_3)$	[redacted]	✓ (3/3)	[redacted]	low
$\mathbf{E}(x_1,x_2) \wedge \mathbf{C}(x_2,x_3) \wedge \mathbf{C}(x_1,x_3)$	[redacted]	✓ (3/3)	[redacted]	low
$\mathbf{E}(x_1,x_2) \wedge \mathbf{E}(x_2,x_3) \wedge \mathbf{C}(x_1,x_3)$	[redacted]	✗ (0/3)	[redacted]	high

Practical concerns: ensure consistency loss doesn't overwhelm your prediction loss; many tricks for this (loss weighting λ, annealing).

Does this Help?

- Li et al. (2019) apply to NLI, focus on basic order relations between inference types (transitivity, symmetry), study different t-norm approaches.



Does this Help?

- **NLP:** NLI (Minervini and Riedel, 2018; Li et al., 2019), question-answering (Asai and Hajishirzi, 2020), relation extraction (Rocktäschel et al., 2015), other (Grespan et al., 2021).

Does this Help?

- **NLP:** NLI (Minervini and Riedel, 2018; Li et al., 2019), question-answering (Asai and Hajishirzi, 2020), relation extraction (Rocktäschel et al., 2015), other (Grespan et al., 2021).

  Can significantly improve consistency and training efficiency, mixed results on improving end-task performance, though.

Does this Help?

- **NLP:** NLI (Minervini and Riedel, 2018; Li et al., 2019), question-answering (Asai and Hajishirzi, 2020), relation extraction (Rocktäschel et al., 2015), other (Grespan et al., 2021).

  Can significantly improve consistency and training efficiency, mixed results on improving end-task performance, though.

- Widely used elsewhere in neural-symbolic modeling (see Marra et al. (2021)), many open technical issues (see Grespan et al. (2021)).

An Alternative Tool: Weighted Model Counting

An Alternative Tool: Weighted Model Counting

(The proper way to do things!)

A Probabilistic Approach

- The semantics of Fuzzy logic has issues, not easy to translate back to ordinary Boolean logic, not amenable to probabilistic inference.

Boolean Propositions	$\mathbf{P} \in \{0, 1\}$
(Probabilistic) Boolean Propositions	$\mathbf{P}$ is 1 with prob. $p \in [0, 1]$ (0 with prob. $1 - p$)
Real-Valued (Fuzzy) Propositions	$\mathbf{P} \in [0, 1]$, *truth degree* $t(\mathbf{P})$

A Probabilistic Approach

- The semantics of Fuzzy logic has issues, not easy to translate back to ordinary Boolean logic, not amenable to probabilistic inference.

Boolean Propositions	$\mathbf{P} \in \{0, 1\}$
(Probabilistic) Boolean Propositions	$\mathbf{P}$ is 1 with prob. $p \in [0, 1]$ (0 with prob. $1 - p$)
Real-Valued (Fuzzy) Propositions	$\mathbf{P} \in [0, 1]$, *truth degree* $t(\mathbf{P})$

Example: Assume we have a proposition $\mathbf{P}$ with a weight 0.3 and we want to get a weight for $\mathbf{P} \wedge \mathbf{P}$ (see (Marra et al., 2021, p43))

A Probabilistic Approach

- The semantics of Fuzzy logic has issues, not easy to translate back to ordinary Boolean logic, not amenable to probabilistic inference.

Boolean Propositions	$\mathbf{P} \in \{0,1\}$
(Probabilistic) Boolean Propositions	$\mathbf{P}$ is 1 with prob. $p \in [0,1]$ (0 with prob. $1-p$)
Real-Valued (Fuzzy) Propositions	$\mathbf{P} \in [0,1]$, *truth degree* $t(\mathbf{P})$

Example: Assume we have a proposition $\mathbf{P}$ with a weight 0.3 and we want to get a weight for $\mathbf{P} \wedge \mathbf{P}$ (see (Marra et al., 2021, p43))

$$p(\mathbf{P} \wedge \mathbf{P}) = p(\mathbf{P}) \qquad \textit{(possible world semantics)}$$

$$t(\mathbf{P} \wedge \mathbf{P}) = t(\mathbf{P}) \times t(\mathbf{P}) = 0.15 \qquad \textit{(product t-norm semantics)}$$

A Probabilistic Approach: Possible World Semantics

Three propositions: $\mathbf{0.9} :: \mathbf{P}_1$ (*A dog is a type of mammal*), $\mathbf{0.8} :: \mathbf{P}_2$ (*A mammal is a type of animal*), $\mathbf{0.45} :: \mathbf{P}_3$ *(A bulldog is a dog)*

world W	$\mathbf{P_1}$	$\mathbf{P_2}$	$\mathbf{P_3}$	$p(W)$
W_1	0	0	0	$(0.1 \times 0.2 \times 0.55) = 0.01$
W_2	1	1	1	$(0.9 \times 0.8 \times 0.45) = 0.32$
W_3	0	0	1	$(0.1 \times 0.2 \times 0.45) = 0.009$
W_4	0	1	1	$(0.1 \times 0.8 \times 0.45) = 0.036$
W_5	1	0	1	$(0.9 \times 0.2 \times 0.45) = 0.081$
W_6	1	1	0	$(0.9 \times 0.8 \times 0.55) = 0.39$
W_7	0	1	0	$(0.1 \times 0.8 \times 0.55) = 0.04$
W_8	1	0	0	$(0.9 \times 0.2 \times 0.55) = 0.08$

A Probabilistic Approach: Possible World Semantics

Three propositions: $\mathbf{0.9} :: \mathbf{P}_1$ (*A dog is a type of mammal*), $\mathbf{0.8} :: \mathbf{P}_2$ (*A mammal is a type of animal*), $\mathbf{0.45} :: \mathbf{P}_3$ *(A bulldog is a dog)*

world W	$\mathbf{P_1}$	$\mathbf{P_2}$	$\mathbf{P_3}$	$p(W)$
W_1	0	0	0	$(0.1 \times 0.2 \times 0.55) = 0.01$
W_2	1	1	1	$(0.9 \times 0.8 \times 0.45) = 0.32$
W_3	0	0	1	$(0.1 \times 0.2 \times 0.45) = 0.009$
W_4	0	1	1	$(0.1 \times 0.8 \times 0.45) = 0.036$
W_5	1	0	1	$(0.9 \times 0.2 \times 0.45) = 0.081$
W_6	1	1	0	$(0.9 \times 0.8 \times 0.55) = 0.39$
W_7	0	1	0	$(0.1 \times 0.8 \times 0.55) = 0.04$
W_8	1	0	0	$(0.9 \times 0.2 \times 0.55) = 0.08$

$$p(W) = \underbrace{\prod_{\mathbf{P} \in W^1} p(\mathbf{P})}_{\text{true in } W,\ W^1} \times \underbrace{\prod_{\mathbf{P} \in W^0} 1 - p(\mathbf{P})}_{\text{false in } W,\ W^0}$$

$$p_{query}(\mathbf{P}) = \sum_{W \text{ s.t. } W \vDash \mathbf{P}} p(W)$$

A Probabilistic Approach: Possible World Semantics

Three propositions: $\mathbf{0.9} :: \mathbf{P}_1$ (*A dog is a type of mammal*), $\mathbf{0.8} :: \mathbf{P}_2$ (*A mammal is a type of animal*), $\mathbf{0.45} :: \mathbf{P}_3$ *(A bulldog is a dog)*

world W	$\mathbf{P_1}$	$\mathbf{P_2}$	$\mathbf{P_3}$	$p(W)$
W_1	0	0	0	$(0.1 \times 0.2 \times 0.55) = 0.01$
W_2	1	1	1	$(0.9 \times 0.8 \times 0.45) = 0.32$
W_3	0	0	1	$(0.1 \times 0.2 \times 0.45) = 0.009$
W_4	0	1	1	$(0.1 \times 0.8 \times 0.45) = 0.036$
W_5	1	0	1	$(0.9 \times 0.2 \times 0.45) = 0.081$
W_6	1	1	0	$(0.9 \times 0.8 \times 0.55) = 0.39$
W_7	0	1	0	$(0.1 \times 0.8 \times 0.55) = 0.04$
W_8	1	0	0	$(0.9 \times 0.2 \times 0.55) = 0.08$

$$p(W) = \underbrace{\prod_{\mathbf{P} \in W^1} p(\mathbf{P})}_{\text{true in } W,\ W^1} \times \underbrace{\prod_{\mathbf{P} \in W^0} 1 - p(\mathbf{P})}_{\text{false in } W,\ W^0}$$

$$p_{query}(\mathbf{P}) = \sum_{W \text{ s.t. } W \models \mathbf{P}} p(W)$$

A Probabilistic Approach: Possible World Semantics

Three propositions: $\mathbf{0.9} :: \mathbf{P}_1$ (*A dog is a type of mammal*), $\mathbf{0.8} :: \mathbf{P}_2$ (*A mammal is a type of animal*), $\mathbf{0.45} :: \mathbf{P}_3$ *(A bulldog is a dog)*

world W	$\mathbf{P_1}$	$\mathbf{P_2}$	$\mathbf{P_3}$	$p(W)$
W_1	0	0	0	$(0.1 \times 0.2 \times 0.55) = 0.01$
W_2	1	1	1	$(0.9 \times 0.8 \times 0.45) = 0.32$
W_3	0	0	1	$(0.1 \times 0.2 \times 0.45) = 0.009$
W_4	0	1	1	$(0.1 \times 0.8 \times 0.45) = 0.036$
W_5	1	0	1	$(0.9 \times 0.2 \times 0.45) = 0.081$
W_6	1	1	0	$(0.9 \times 0.8 \times 0.55) = 0.39$
W_7	0	1	0	$(0.1 \times 0.8 \times 0.55) = 0.04$
W_8	1	0	0	$(0.9 \times 0.2 \times 0.55) = 0.08$

$$p(W) = \underbrace{\prod_{\mathbf{P} \in W^1} p(\mathbf{P})}_{\text{true in } W,\ W^1} \times \underbrace{\prod_{\mathbf{P} \in W^0} 1 - p(\mathbf{P})}_{\text{false in } W,\ W^0}$$

$$p_{query}(\mathbf{P}) = \sum_{W \text{ s.t. } W \models \mathbf{P}} p(W)$$

A Probabilistic Approach: Weighted Model Counting

- **Querying**: generalizes to any propositional formula α and reducible to the problem of Weighted Model Counting (WMC) (Chavira and Darwiche, 2008)

$$p_{query}(\alpha) = \underbrace{\sum_{W \text{ s.t. } W \models \alpha} p(W)}_{\text{WMC}}$$

A Probabilistic Approach: Weighted Model Counting

- **Querying**: generalizes to any propositional formula α and reducible to the problem of Weighted Model Counting (WMC) (Chavira and Darwiche, 2008)

$$p_{query}(\alpha) = \underbrace{\sum_{W \text{ s.t. } W \models \alpha} p(W)}_{\text{WMC}}$$

constraints

$$\boxed{\mathbf{P}_1 \rightarrow \mathbf{P}_2 \wedge \mathbf{P}_2 \rightarrow \mathbf{P}_3}$$

world W	$\mathbf{P_1}$	$\mathbf{P_2}$	$\mathbf{P_3}$	model?	$p(W)$
W_1	0	0	0	yes	$(0.1 \times 0.2 \times 0.55) = 0.01$
W_2	1	1	1	yes	$(0.9 \times 0.8 \times 0.45) = 0.32$
W_3	0	0	1	yes	$(0.1 \times 0.2 \times 0.45) = 0.009$
W_4	0	1	1	yes	$(0.1 \times 0.8 \times 0.45) = 0.036$
W_5	1	0	1	no	0.0
W_6	1	1	0	no	0.0
W_7	0	1	0	no	0.0
W_8	1	0	0	no	0.0
				#SAT : 4	WMC: 0.375
					$p(\mathbf{P}_3) = 0.365/(1 - 0.365)$

A Probabilistic Approach: Weighted Model Counting

- **Querying**: generalizes to any propositional formula α and reducible to the problem of Weighted Model Counting (WMC) (Chavira and Darwiche, 2008)

$$p_{query}(\alpha) = \underbrace{\sum_{W \text{ s.t. } W \models \alpha} p(W)}_{\text{WMC}}$$

constraints

$$\boxed{\alpha = \mathbf{P}_1 \rightarrow \mathbf{P}_2 \wedge \mathbf{P}_2 \rightarrow \mathbf{P}_3}$$

world W	$\mathbf{P_1}$	$\mathbf{P_2}$	$\mathbf{P_3}$	model?	$p(W)$
W_1	0	0	0	yes	$(0.1 \times 0.2 \times 0.55) = 0.01$
W_2	1	1	1	yes	$(0.9 \times 0.8 \times 0.45) = 0.32$
W_3	0	0	1	yes	$(0.1 \times 0.2 \times 0.45) = 0.009$
W_4	0	1	1	yes	$(0.1 \times 0.8 \times 0.45) = 0.036$
W_5	1	0	1	yes	$(0.9 \times 0.2 \times 0.45) = 0.081$
W_6	1	1	0	yes	$(0.9 \times 0.8 \times 0.55) = 0.39$
W_7	0	1	0	yes	$(0.1 \times 0.8 \times 0.55) = 0.04$
W_8	1	0	0	yes	$(0.9 \times 0.2 \times 0.55) = 0.08$

$p(\alpha) = 0.375$

A Probabilistic Approach: Weighted Model Counting

- **Querying**: generalizes to any propositional formula α and reducible to the problem of Weighted Model Counting (WMC) (Chavira and Darwiche, 2008)

$$p_{query}(\alpha) = \underbrace{\sum_{W \text{ s.t. } W \models \alpha} p(W)}_{\text{WMC}}$$

constraints

$$\boxed{\alpha = \mathbf{P}_1 \rightarrow \mathbf{P}_2 \wedge \mathbf{P}_2 \rightarrow \mathbf{P}_3}$$

world W	$\mathbf{P_1}$	$\mathbf{P_2}$	$\mathbf{P_3}$	model?	$p(W)$
W_1	0	0	0	yes	$(0.1 \times 0.2 \times 0.55) = 0.01$
W_2	1	1	1	yes	$(0.9 \times 0.8 \times 0.45) = 0.32$
W_3	0	0	1	yes	$(0.1 \times 0.2 \times 0.45) = 0.009$
W_4	0	1	1	yes	$(0.1 \times 0.8 \times 0.45) = 0.036$
W_5	1	0	1	yes	$(0.9 \times 0.2 \times 0.45) = 0.081$
W_6	1	1	0	yes	$(0.9 \times 0.8 \times 0.55) = 0.39$
W_7	0	1	0	yes	$(0.1 \times 0.8 \times 0.55) = 0.04$
W_8	1	0	0	yes	$(0.9 \times 0.2 \times 0.55) = 0.08$

$p(\alpha) = 0.375$

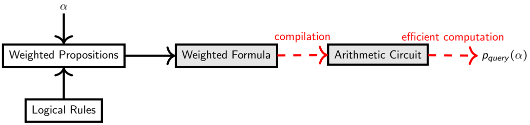
Other: **MARG** or **SUCC** inference, *query probability)*, (*WMC*), **MPE**/**MAP**, most likely world (*MaxSAT*) (De Raedt and Kimmig, 2015).

A Probabilistic Approach: Weighted Model Counting

- **Querying**: generalizes to any propositional formula α and reducible to the problem of Weighted Model Counting (WMC) (Raedt et al., 2016)

$$p_{query}(\alpha) = \underbrace{\sum_{W \text{ s.t. } W \vDash \alpha} p(W)}_{\text{WMC}}$$

Efficient marginal computations through knowledge compilation (Darwiche and Marquis, 2002), many open-source compilers and tools.

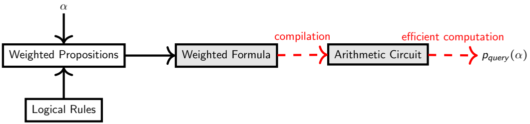


A Probabilistic Approach: Weighted Model Counting

- **Querying**: generalizes to any propositional formula α and reducible to the problem of Weighted Model Counting (WMC) (Raedt et al., 2016)

$$p_{query}(\alpha) = \underbrace{\sum_{W \text{ s.t. } W \models \alpha} p(W)}_{\text{WMC}}$$

Efficient marginal computations through knowledge compilation (Darwiche and Marquis, 2002), many open-source compilers and tools.




```
from pysdd.sdd import SddManager, Vtree, WmcManager # pip install PySDD
vtree = Vtree(var_count=4, var_order=[2,1,4,3], vtree_type="balanced")
sdd = SddManager.from_vtree(vtree); a, b, c, d = sdd.vars

alpha = (a & b) | (b & c) | (c & d)
wmc = alpha.(log_mode=False); wmc.set_literal_weight(a, 0.5)
print(f"Weighted Model Count: {wmc.propagate()}")
```

Logic as Loss Function: probabilistic variant

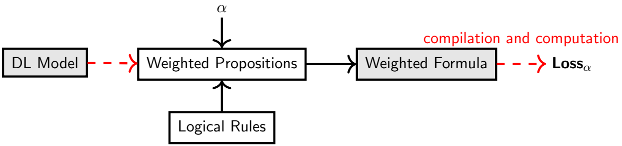
- Provides an alternative to fuzzy semantics, compute *marginal probabilities* of formulas α over propositions $\mathbf{P}_1, .., \mathbf{P}_n$

$$\mathbf{Loss}_\alpha \propto -\log \sum_{W \text{ s.t. } W \models \alpha} \underbrace{\prod_{\mathbf{P} \in W^1} p_{model}(\mathbf{P}) \cdot \prod_{\mathbf{P} \in W^0} 1 - p_{model}(\mathbf{P})}_{\text{parameterized by our model}}$$

Logic as Loss Function: probabilistic variant

- Provides an alternative to fuzzy semantics, compute *marginal probabilities* of formulas α over propositions $\mathbf{P}_1, .., \mathbf{P}_n$

$$\mathbf{Loss}_\alpha \propto -\log \sum_{W \text{ s.t. } W \models \alpha} \underbrace{\prod_{\mathbf{P} \in W^1} p_{model}(\mathbf{P}) \cdot \prod_{\mathbf{P} \in W^0} 1 - p_{model}(\mathbf{P})}_{\text{parameterized by our model}}$$



Logic as Loss Function: probabilistic variant

- Provides an alternative to fuzzy semantics, compute *marginal probabilities* of formulas α over propositions $\mathbf{P}_1, .., \mathbf{P}_n$

$$\mathbf{Loss}_\alpha \propto -\log \sum_{W \text{ s.t. } W \models \alpha} \underbrace{\prod_{\mathbf{P} \in W^1} p_{model}(\mathbf{P}) \cdot \prod_{\mathbf{P} \in W^0} 1 - p_{model}(\mathbf{P})}_{\text{parameterized by our model}}$$

semantic loss function (Xu et al., 2018)

> The semantic loss is proportional to a negative logarithm of the probability of generating a state that satisfies the constraint when sampling values according to p. Hence, it is the self-information (or 'surprise') of obtaining an assignment that satisfies the constraint...

Logic as Loss Function: probabilistic variant

- Provides an alternative to fuzzy semantics, compute *marginal probabilities* of formulas α over propositions $\mathbf{P}_1, .., \mathbf{P}_n$

$$\mathbf{Loss}_\alpha \propto -\log \sum_{W \text{ s.t. } W \models \alpha} \underbrace{\prod_{\mathbf{P} \in W^1} p_{model}(\mathbf{P}) \cdot \prod_{\mathbf{P} \in W^0} 1 - p_{model}(\mathbf{P})}_{\text{parameterized by our model}}$$

semantic loss function (Xu et al., 2018)

> The semantic loss is proportional to a negative logarithm of the probability of generating a state that satisfies the constraint when sampling values according to p. Hence, it is the self-information (or 'surprise') of obtaining an assignment that satisfies the constraint...

As before, often used as *undirected* model (alternative (Manhaeve et al., 2018)):

$$\mathbf{Loss} = \underbrace{\mathbf{Loss}_{\text{prediction}}}_{\text{ordinary loss}} + \underbrace{\lambda \mathbf{Loss}_\alpha}_{\text{logical constraints}}$$

The NLI Example Again

x1: A man with a hat is riding his bicycle down the street.
x2: A person is moving with the help of their legs.
Prediction: **Entailment**

x2: A person is moving with the help of their legs.
x3: A person is moving
Prediction: **Entailment**

$$\underbrace{0.95 :: \mathbf{E}(x_1, x_2) \land 0.85 :: \mathbf{E}(x_2, x_3) \land 0.75 :: \mathbf{C}(x_1, x_3) \equiv \neg\mathbf{E}(x_1, x_3)}_{\textbf{(weighted) model predictions}} \longrightarrow$$

$\alpha_{\text{constraints}}$ (before grounding)

$$\forall x, y, z(\mathbf{E}(x, y) \land \mathbf{E}(y, z) \rightarrow \mathbf{E}(x, z))$$
$$\forall x, y, z(\mathbf{E}(x, y) \land \mathbf{C}(y, z) \rightarrow \mathbf{C}(x, z))$$
$$\forall x, y(\mathbf{C}(x, y) \rightarrow \mathbf{C}(y, x))$$

world W	$\mathbf{E}(x_1, x_2)$	$\mathbf{E}(x_2, x_3)$	$\mathbf{E}(x_1, x_3)$	$W \models \alpha$	$p(W)$
W_1	0	0	0	yes	$(0.05 \times 0.15 \times 0.75) = 0.005$
W_2	1	1	1	yes	$(0.95 \times 0.85 \times 0.25) = 0.201$
W_3	0	0	1	yes	$(0.05 \times 0.15 \times 0.25) = 0.001$
W_4	0	1	1	yes	$(0.05 \times 0.85 \times 0.25) = 0.010$
W_5	1	0	1	no	$(0.95 \times 0.15 \times 0.25) = 0.03$
W_6	1	1	0	no	$(0.95 \times 0.85 \times 0.75) = 0.60$
W_7	0	1	0	yes	$(0.05 \times 0.85 \times 0.75) = 0.03$
W_8	1	0	0	yes	$(0.95 \times 0.15 \times 0.75) = 0.10$

$p(\alpha_{\text{constraints}}) \approx 0.34$

The NLI Example Again

x1: A man with a hat is riding his bicycle down the street.
x2: A person is moving with the help of their legs.
Prediction: **Entailment**

x2: A person is moving with the help of their legs.
x3: A person is moving
Prediction: **Entailment**

$\alpha_{\text{constraints}}$ (before grounding)

$$\underbrace{0.95 :: \mathbf{E}(x_1, x_2) \wedge 0.85 :: \mathbf{E}(x_2, x_3) \wedge 0.75 :: \mathbf{C}(x_1, x_3) \equiv \neg\mathbf{E}(x_1, x_3)}_{\text{(weighted) model predictions, } \alpha_{\text{pred}} = \mathbf{E}(x_1,x_2) \wedge \mathbf{E}(x_2,x_3) \wedge \mathbf{C}(x_1,x_3)} \longrightarrow$$

$$\forall x, y, z(\mathbf{E}(x, y) \wedge \mathbf{E}(y, z) \rightarrow \mathbf{E}(x, z))$$
$$\forall x, y, z(\mathbf{E}(x, y) \wedge \mathbf{C}(y, z) \rightarrow \mathbf{C}(x, z))$$
$$\forall x, y(\mathbf{C}(x, y) \rightarrow \mathbf{C}(y, x))$$

world W	$\mathbf{E}(x_1, x_2)$	$\mathbf{E}(x_2, x_3)$	$\mathbf{E}(x_1, x_3)$	$W \vDash \alpha$	$p(W)$
W_1	0	0	0	no	$(0.05 \times 0.15 \times 0.75) = 0.005$
W_2	1	1	1	no	$(0.95 \times 0.85 \times 0.25) = 0.201$
W_3	0	0	1	no	$(0.05 \times 0.15 \times 0.25) = 0.001$
W_4	0	1	1	no	$(0.05 \times 0.85 \times 0.25) = 0.010$
W_5	1	0	1	no	$(0.95 \times 0.15 \times 0.25) = 0.03$
W_6	1	1	0	no	**violates constraints**
W_7	0	1	0	no	$(0.05 \times 0.85 \times 0.75) = 0.03$
W_8	1	0	0	no	$(0.95 \times 0.15 \times 0.75) = 0.10$

$p(\alpha_{\text{constraints}} \wedge \alpha_{\text{pred}}) = 0.0$

The NLI Example Again

x1: A man with a hat is riding his bicycle down the street.
x2: A person is moving with the help of their legs.
Prediction: **Entailment**

x2: A person is moving with the help of their legs.
x3: A person is moving
Prediction: **Entailment**

$$\underbrace{0.95 :: \mathbf{E}(x_1, x_2) \wedge 0.85 :: \mathbf{E}(x_2, x_3) \wedge 0.75 :: \mathbf{C}(x_1, x_3)}_{\textbf{(weighted) model predictions, } \alpha_{\text{pred}} = \mathbf{E}(x_1,x_2) \wedge \mathbf{E}(x_2,x_3) \wedge \mathbf{C}(x_1,x_3)} \equiv \neg \mathbf{E}(x_1, x_3) \rightarrow$$

$$\forall x, y, z(\mathbf{E}(x, y) \wedge \mathbf{E}(y, z) \rightarrow \mathbf{E}(x, z))$$
$$\forall x, y, z(\mathbf{E}(x, y) \wedge \mathbf{C}(y, z) \rightarrow \mathbf{C}(x, z))$$
$$\forall x, y(\mathbf{C}(x, y) \rightarrow \mathbf{C}(y, x))$$

world W	$\mathbf{E}(x_1, x_2)$	$\mathbf{E}(x_2, x_3)$	$\mathbf{E}(x_1, x_3)$	$W \models \alpha$	$p(W)$
W_1	0	0	0	no	$(0.05 \times 0.15 \times 0.75) = 0.005$
W_2	1	1	1	no	$(0.95 \times 0.85 \times 0.25) = 0.201$
W_3	0	0	1	no	$(0.05 \times 0.15 \times 0.25) = 0.001$
W_4	0	1	1	no	$(0.05 \times 0.85 \times 0.25) = 0.010$
W_5	1	0	1	no	$(0.95 \times 0.15 \times 0.25) = 0.03$
W_6	1	1	0	yes	$(0.95 \times 0.85 \times 0.75) = 0.60$
W_7	0	1	0	no	$(0.05 \times 0.85 \times 0.75) = 0.03$
W_8	1	0	0	no	$(0.95 \times 0.15 \times 0.75) = 0.10$

$p(\alpha_{\text{pred}}) = 0.60$

The NLI Example Again

x1: A man with a hat is riding his bicycle down the street.
x2: A person is moving with the help of their legs.
Prediction: **Entailment**

x2: A person is moving with the help of their legs.
x3: A person is moving
Prediction: **Entailment**

$$\underbrace{0.95 :: \mathbf{E}(x_1, x_2) \wedge 0.85 :: \mathbf{E}(x_2, x_3) \wedge 0.75 :: \mathbf{C}(x_1, x_3)}_{\textbf{(weighted) model predictions, } \alpha_{\text{pred}} = \mathbf{E}(x_1,x_2) \wedge \mathbf{E}(x_2,x_3) \wedge \mathbf{C}(x_1,x_3)} \equiv \neg \mathbf{E}(x_1, x_3) \longrightarrow$$

$$\forall x, y, z(\mathbf{E}(x, y) \wedge \mathbf{E}(y, z) \rightarrow \mathbf{E}(x, z))$$
$$\forall x, y, z(\mathbf{E}(x, y) \wedge \mathbf{C}(y, z) \rightarrow \mathbf{C}(x, z))$$
$$\forall x, y(\mathbf{C}(x, y) \rightarrow \mathbf{C}(y, x))$$

world W	$\mathbf{E}(x_1, x_2)$	$\mathbf{E}(x_2, x_3)$	$\mathbf{E}(x_1, x_3)$	$W \vDash \alpha$	$p(W)$
W_1	0	0	0	no	$(0.05 \times 0.15 \times 0.75) = 0.005$
W_2	1	1	1	no	$(0.95 \times 0.85 \times 0.25) = 0.201$
W_3	0	0	1	no	$(0.05 \times 0.15 \times 0.25) = 0.001$
W_4	0	1	1	no	$(0.05 \times 0.85 \times 0.25) = 0.010$
W_5	1	0	1	no	$(0.95 \times 0.15 \times 0.25) = 0.03$
W_6	1	1	0	yes	$(0.95 \times 0.85 \times 0.75) = 0.60$
W_7	0	1	0	no	$(0.05 \times 0.85 \times 0.75) = 0.03$
W_8	1	0	0	no	$(0.95 \times 0.15 \times 0.75) = 0.10$

$p(\alpha_{\text{pred}}) = 0.60$

Ordinary loss is again special case: $-\log p(\alpha_{\text{pred}}) = \mathbf{Loss}_{\text{ordinary loss}}$

Conclusion

- **Neural Symbolic modeling**, focusing on the ‘logic as loss function‘ and ‘logic in weights‘ approaches, **undirected models**

  conceptual tools connecting logic and DL: thinking of model predictions as symbolic objects, annotations as logical specifications

  technical tools: fuzzy and soft logic relaxations, model counting (probabilistic approach), translating logic to loss functions.

Still a niche area in NLP, many exciting topics to explore.

Credits and Additional Reading

- Many ideas and examples taken from the following (beyond what's cited):

  Guy Van den Broeck et al. tutorial: https://web.cs.ucla.edu/~guyvdb/talks/IJCAI16-tutorial/, (Fierens et al., 2015; Raedt et al., 2016; Manhaeve et al., 2021),

  Additional resources: *Problog*: https://dtai.cs.kuleuven.be/problog/, *PySDD*: https://github.com/wannesm/PySDD, *Pylon*: https://pylon-lib.github.io/

Thank you.

References I

Asai, A. and Hajishirzi, H. (2020). Logic-guided data augmentation and regularization for consistent question answering. *arXiv preprint arXiv:2004.10157*.

Chavira, M. and Darwiche, A. (2008). On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6-7):772–799.

Darwiche, A. and Marquis, P. (2002). A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264.

De Raedt, L. and Kimmig, A. (2015). Probabilistic (logic) programming concepts. *Machine Learning*, 100(1):5–47.

Fierens, D., Van den Broeck, G., Renkens, J., Shterionov, D., Gutmann, B., Thon, I., Janssens, G., and De Raedt, L. (2015). Inference and learning in probabilistic logic programs using weighted boolean formulas. *Theory and Practice of Logic Programming*, 15(3):358–401.

Grespan, M. M., Gupta, A., and Srikumar, V. (2021). Evaluating relaxations of logic for neural networks: A comprehensive study. *arXiv preprint arXiv:2107.13646*.

Li, T., Gupta, V., Mehta, M., and Srikumar, V. (2019). A logic-driven framework for consistency of neural models. *arXiv preprint arXiv:1909.00126*.

Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., and De Raedt, L. (2018). Deepproblog: Neural probabilistic logic programming. *Advances in Neural Information Processing Systems*, 31.

References II

Manhaeve, R., Dumančić, S., Kimmig, A., Demeester, T., and De Raedt, L. (2021). Neural probabilistic logic programming in deepproblog. *Artificial Intelligence*, 298:103504.

Marra, G., Dumančić, S., Manhaeve, R., and De Raedt, L. (2021). From statistical relational to neural symbolic artificial intelligence: a survey. *arXiv preprint arXiv:2108.11451*.

Minervini, P. and Riedel, S. (2018). Adversarially regularising neural nli models to integrate logical background knowledge. *arXiv preprint arXiv:1808.08609*.

Raedt, L. D., Kersting, K., Natarajan, S., and Poole, D. (2016). Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis lectures on artificial intelligence and machine learning*, 10(2):1–189.

Rocktäschel, T., Singh, S., and Riedel, S. (2015). Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of the 2015 conference of the north American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1119–1129.

Xu, J., Zhang, Z., Friedman, T., Liang, Y., and Broeck, G. (2018). A semantic loss function for deep learning with symbolic knowledge. In *International conference on machine learning*, pages 5502–5511. PMLR.