Logic as Loss: Semantic loss

THE UNIVERSITY OF UTAH

The idea of the "logic as loss" framework

What we want of our models

Satisfy a set of invariant properties
(perhaps including properties about specific examples)

Minimize Constraint loss

Let us formally state the setting

Suppose we have a sentence α in predicate logic, defined over some atoms $X = \{X_1, X_2, \cdots, X_n\}$

Suppose each atom X_i is associated with a probability p_i, possibly from a neural model

Let the vector $\mathbf{p}$ denote the collection of probabilities $[p_1, p_2, \cdots, p_n]$ over the atoms

Our goal:

To define a loss function $L(\alpha, \mathbf{p})$ such that minimizing it produces a model (and associated probabilities) that assigns labels satisfying the sentence α

The idea of the “logic as loss” framework

What we want of our models

Satisfy a set of invariant properties
(perhaps including properties about specific examples)

Minimize Constraint loss

What are some desirable properties of the loss function $L(\alpha, p)$?

Let us formally state the setting

Suppose we have a sentence α in predicate logic, defined over some atoms $X = \{X_1, X_2, \cdots, X_n\}$

Suppose each atom X_i is associated with a probability p_i, possibly from a neural model

Let the vector $\mathbf{p}$ denote the collection of probabilities $[p_1, p_2, \cdots, p_n]$ over the atoms

Our goal:

To define a loss function $L(\alpha, \mathbf{p})$ such that minimizing it produces a model (and associated probabilities) that assigns labels satisfying the sentence α

What are some desirable properties of the loss function $L(\alpha, \mathbf{p})$?

Let us brainstorm

What are some desirable properties of the loss function $L(\alpha, \mathbf{p})$?

Let us brainstorm

Semantic loss: An axiomatic approach

Xu, Jingyi, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. "A semantic loss function for deep learning with symbolic knowledge." In *ICML 2018*

Logic as loss: Semantic loss

- Building up to semantic loss: The axioms
- Semantic loss
- Examples
  - Conjunction
  - Implication
- Complex constraints & Weighted Model Counting
  - Example: The exactly-one constraint

Logic as loss: Semantic loss

- Building up to semantic loss: The axioms

- Semantic loss

- Examples
  - Conjunction
  - Implication

- Complex constraints & Weighted Model Counting
  - Example: The exactly-one constraint

The Differentiability Axiom

We want to be able to take gradients of the loss function

The differentiability axiom: For any fixed α, the semantic loss $\mathbf{L}(\alpha, \mathbf{p})$ is monotone in each probability in $\mathbf{p}$, and is differentiable.

What are some desirable properties of the loss function $L(\alpha, \mathbf{p})$?

1. The loss should be sub-differentiable

Xu, Jingyi, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. "A semantic loss function for deep learning with symbolic knowledge." In *ICML 2018*

Logical entailments

Suppose we have two sentences in logic α and β such that $\alpha \vDash \beta$

Logical entailments

Suppose we have two sentences in logic α and β such that $\alpha \vDash \beta$

Some examples:

$\alpha = X_1 \wedge X_2$ and $\beta = X_1$

$\alpha = X_1 \wedge X_2$ and $\beta = X_1 \rightarrow X_2$

Logical entailments

Suppose we have two sentences in logic α and β such that $\alpha \vDash \beta$

Some examples:

$\alpha = X_1 \wedge X_2$ and $\beta = X_1$
$\alpha = X_1 \wedge X_2$ and $\beta = X_1 \to X_2$

In each case, whenever α is `true`, so is β. That is $\alpha \vDash \beta$.

Logical entailments

Suppose we have two sentences in logic α and β such that $\alpha \vDash \beta$

Some examples:

$\alpha = X_1 \wedge X_2$ and $\beta = X_1$
$\alpha = X_1 \wedge X_2$ and $\beta = X_1 \rightarrow X_2$

In each case, whenever α is `true`, so is β. That is $\alpha \vDash \beta$.

Can we expect anything about the relative values of the losses $L(\alpha, p)$ and $L(\beta, p)$ associated with these sentences, *irrespective of the value of* p?

Logical entailments

Suppose we have two sentences in logic α and β such that $\alpha \vDash \beta$

Some examples:

$\alpha = X_1 \wedge X_2$ and $\beta = X_1$
$\alpha = X_1 \wedge X_2$ and $\beta = X_1 \rightarrow X_2$

In each case, whenever α is `true`, so is β. That is $\alpha \vDash \beta$.

Can we expect anything about the relative values of the losses $L(\alpha, p)$ and $L(\beta, p)$ associated with these sentences, *irrespective of the value of* p?

Intuition: Since α is a stricter condition than β, violating it should face a stronger penalty. That is, we would like $L(\alpha, p) \geq L(\beta, p)$.

The Monotonicity Axiom

Suppose we have two sentences in logic α and β such that $\alpha \vDash \beta$

Some examples:

$\alpha = X_1 \wedge X_2$ and $\beta = X_1$
$\alpha = X_1 \wedge X_2$ and $\beta = X_1 \rightarrow X_2$

In each case, whenever α is `true`, so is β. That is $\alpha \vDash \beta$.

Can we expect anything about the relative values of the losses $L(\alpha, p)$ and $L(\beta, p)$ associated with these sentences, *irrespective of the value of* p?

Intuition: Since α is a stricter condition than β, violating it should face a stronger penalty. That is, we would like $L(\alpha, p) \geq L(\beta, p)$.

The monotonicity axiom: *If* $\alpha \vDash \beta$*, then* $L(\alpha, p) \geq L(\beta, p)$ *for any value of* p*.*

What are some desirable properties of the loss function $L(\alpha, \mathbf{p})$?

1. The loss should be sub-differentiable

2. For two sentences α and β, if $\alpha \vDash \beta$, we want $L(\alpha, \mathbf{p}) \geq L(\beta, \mathbf{p})$

Xu, Jingyi, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. "A semantic loss function for deep learning with symbolic knowledge." In *ICML 2018*

A consequence of the monotonicity axiom

Consider two sentences α and β that are syntactically different, but semantically the same

A consequence of the monotonicity axiom

Consider two sentences α and β that are syntactically different, but semantically the same

Some examples:

$\alpha = X_1 \rightarrow X_2$ and $\beta = \neg X_1 \vee X_2$

$\alpha = X_1 \rightarrow (X_2 \rightarrow X_3)$ and $\beta = \neg X_3 \rightarrow \neg(X_1 \wedge X_2)$

A consequence of the monotonicity axiom

Consider two sentences α and β that are syntactically different, but semantically the same

Some examples:

$\alpha = X_1 \rightarrow X_2$ and $\beta = \neg X_1 \vee X_2$

$\alpha = X_1 \rightarrow (X_2 \rightarrow X_3)$ and $\beta = \neg X_3 \rightarrow \neg(X_1 \wedge X_2)$

X_1	X_2	X_3	$X_1 \rightarrow (X_2 \rightarrow X_3)$	$\neg X_3 \rightarrow \neg(X_1 \wedge X_2)$
$\top$	$\top$	$\top$	$\top$	$\top$
$\top$	$\top$	$\bot$	$\bot$	$\bot$
$\top$	$\bot$	$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\top$	$\top$
$\bot$	$\top$	$\top$	$\top$	$\top$
$\bot$	$\top$	$\bot$	$\top$	$\top$
$\bot$	$\bot$	$\top$	$\top$	$\top$
$\bot$	$\bot$	$\bot$	$\top$	$\top$

A consequence of the monotonicity axiom

Consider two sentences α and β that are syntactically different, but semantically the same

Some examples:

$\alpha = X_1 \to X_2$ and $\beta = \neg X_1 \vee X_2$

$\alpha = X_1 \to (X_2 \to X_3)$ and $\beta = \neg X_3 \to \neg(X_1 \wedge X_2)$

In these cases, we write $\alpha \equiv \beta$ and say "α is logically equivalent to β"

A consequence of the monotonicity axiom

Consider two sentences α and β that are syntactically different, but semantically the same

Some examples:

$\alpha = X_1 \rightarrow X_2$ and $\beta = \neg X_1 \vee X_2$

$\alpha = X_1 \rightarrow (X_2 \rightarrow X_3)$ and $\beta = \neg X_3 \rightarrow \neg(X_1 \wedge X_2)$

In these cases, we write $\alpha \equiv \beta$ and say "α is logically equivalent to β"

both $\alpha \vDash \beta$ and $\beta \vDash \alpha$

A consequence of the monotonicity axiom

Consider two sentences α and β that are syntactically different, but semantically the same

Some examples:

$\alpha = X_1 \rightarrow X_2$ and $\beta = \neg X_1 \vee X_2$

$\alpha = X_1 \rightarrow (X_2 \rightarrow X_3)$ and $\beta = \neg X_3 \rightarrow \neg(X_1 \wedge X_2)$

In these cases, we write $\alpha \equiv \beta$ and say "α is logically equivalent to β"

both $\alpha \vDash \beta$ and $\beta \vDash \alpha$

$\mathrm{L}(\alpha, \mathrm{p}) \geq \mathrm{L}(\beta, \mathrm{p})$
for any value of p

A consequence of the monotonicity axiom

Consider two sentences α and β that are syntactically different, but semantically the same

Some examples:

$\alpha = X_1 \to X_2$ and $\beta = \neg X_1 \lor X_2$

$\alpha = X_1 \to (X_2 \to X_3)$ and $\beta = \neg X_3 \to \neg(X_1 \land X_2)$

In these cases, we write $\alpha \equiv \beta$ and say "α is logically equivalent to β"

both $\alpha \vDash \beta$ and $\beta \vDash \alpha$

$\mathrm{L}(\alpha, \mathrm{p}) \geq \mathrm{L}(\beta, \mathrm{p})$
for any value of $\mathbf{p}$

and

$\mathrm{L}(\beta, \mathrm{p}) \geq \mathrm{L}(\alpha, \mathrm{p})$
for any value of $\mathbf{p}$

A consequence of the monotonicity axiom

Consider two sentences α and β that are syntactically different, but semantically the same

Some examples:

$\alpha = X_1 \rightarrow X_2$ and $\beta = \neg X_1 \vee X_2$

$\alpha = X_1 \rightarrow (X_2 \rightarrow X_3)$ and $\beta = \neg X_3 \rightarrow \neg(X_1 \wedge X_2)$

In these cases, we write $\alpha \equiv \beta$ and say "α is logically equivalent to β"

both $\alpha \vDash \beta$ and $\beta \vDash \alpha$

$\mathrm{L}(\alpha, \mathrm{p}) \geq \mathrm{L}(\beta, \mathrm{p})$ for any value of p and $\mathrm{L}(\beta, \mathrm{p}) \geq \mathrm{L}(\alpha, \mathrm{p})$ for any value of p $\Rightarrow$ $\mathrm{L}(\alpha, \mathrm{p}) = \mathrm{L}(\beta, \mathrm{p})$

Monotonicity Axiom → Semantics

Consider two sentences α and β that are syntactically different, but semantically the same

That is, both $\alpha \vDash \beta$ and $\beta \vDash \alpha$

Then, $\mathrm{L}(\alpha, \mathrm{p}) = \mathrm{L}(\beta, \mathrm{p})$ for any value of p

For logically equivalent statements any loss function that satisfies monotonicity will have equal loss values

In other words, the loss is not affected by syntactic variations in how the constraints are written

Monotonicity Axiom → Semantics

Consider two sentences α and β that are syntactically different, but semantically the same

That is, both $\alpha \vDash \beta$ and $\beta \vDash \alpha$

Then, $\mathrm{L}(\alpha, \mathrm{p}) = \mathrm{L}(\beta, \mathrm{p})$ for any value of $\mathbf{p}$

For logically equivalent statements any loss function that satisfies monotonicity will have equal loss values

In other words, the loss is not affected by syntactic variations in how the constraints are written

Another consequence of monotonicity: $\mathrm{L}(\alpha, \mathrm{p}) > 0$ for any $\alpha, \mathbf{p}$

Exercise: Prove this

What are some desirable properties of the loss function $L(\alpha, \mathbf{p})$?

1. The loss should be sub-differentiable

2. For two sentences α and β, if $\alpha \vDash \beta$, we want $L(\alpha, \mathbf{p}) \geq L(\beta, \mathbf{p})$

   → Logically equivalent sentences should have equal losses

Xu, Jingyi, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. "A semantic loss function for deep learning with symbolic knowledge." In *ICML 2018*

A bit of notation

Suppose we have a set of Boolean variables $X = \{X_1, X_2, \cdots X_n\}$
We have a specific assignment of truth values to these variables, denoted by $x = \{x_1, x_2, \cdots, x_n\}$

We can write this assignment as a binary vector or as a logical sentence

A bit of notation

Suppose we have a set of Boolean variables $X = \{X_1, X_2, \cdots X_n\}$
We have a specific assignment of truth values to these variables, denoted by $x = \{x_1, x_2, \cdots, x_n\}$

We can write this assignment as a binary vector or as a logical sentence

Example: Consider $\{X_1 = \top, X_2 = \top, X_3 = \bot\}$. That is, $x = [\top, \top, \bot]$

A bit of notation

Suppose we have a set of Boolean variables $X = \{X_1, X_2, \cdots X_n\}$
We have a specific assignment of truth values to these variables, denoted by $x = \{x_1, x_2, \cdots, x_n\}$

We can write this assignment as a binary vector or as a logical sentence

Example: Consider $\{X_1 = \mathsf{T}, X_2 = \mathsf{T}, X_3 = \perp\}$. That is, $x = [\mathsf{T}, \mathsf{T}, \perp]$

Binary vector: We can write the assignment as the vector $[1,1,0]$

(We could even interpret these as probabilities that each variable is true)

A bit of notation

Suppose we have a set of Boolean variables $X = \{X_1, X_2, \cdots X_n\}$

We have a specific assignment of truth values to these variables, denoted by $x = \{x_1, x_2, \cdots, x_n\}$

We can write this assignment as a binary vector or as a logical sentence

Example: Consider $\{X_1 = \top, X_2 = \top, X_3 = \bot\}$. That is, $x = [\top, \top, \bot]$

Binary vector: We can write the assignment as the vector $[1,1,0]$

(We could even interpret these as probabilities that each variable is true)

This can be interpreted as a probability that each variable is `true`

A bit of notation

Suppose we have a set of Boolean variables $X = \{X_1, X_2, \cdots X_n\}$
We have a specific assignment of truth values to these variables, denoted by $x = \{x_1, x_2, \cdots, x_n\}$

We can write this assignment as a binary vector or as a logical sentence

Example: Consider $\{X_1 = \top, X_2 = \top, X_3 = \bot\}$. That is, $x = [\top, \top, \bot]$

Binary vector: We can write the assignment as the vector $[1,1,0]$

(We could even interpret these as probabilities that each variable is true)

Logical sentence: We can also write the assignment as the sentence $X_1 \wedge X_2 \wedge \neg X_3$

There is only one assignment that makes this conjunction true, and that is the above one

A bit of notation

Suppose we have a set of Boolean variables $X = \{X_1, X_2, \cdots X_n\}$
We have a specific assignment of truth values to these variables, denoted by $x = \{x_1, x_2, \cdots, x_n\}$

We can write this assignment as a binary vector or as a logical sentence

Example: Consider $\{X_1 = \top, X_2 = \top, X_3 = \bot\}$. That is, $x = [\top, \top, \bot]$

Binary vector: We can write the assignment as the vector $[1,1,0]$

(We could even interpret these as probabilities that each variable is true)

Logical sentence: We can also write the assignment as the sentence $X_1 \wedge X_2 \wedge \neg X_3$

There is only one assignment that makes this conjunction true, and that is the above one

Essentially, all three are saying the same thing

How should the loss handle a perfect match?

Consider an assignment to a set of Boolean variables (also called a *state*)

Let x denote its representation as a binary vector (to be interpreted as a probability)

Let α denote its representation as a logical sentence

What can we say about the loss $L(\alpha, x)$?

How should the loss handle a perfect match?

Consider an assignment to a set of Boolean variables (also called a *state*)

Let x denote its representation as a binary vector (to be interpreted as a probability)

Let α denote its representation as a logical sentence

What can we say about the loss $L(\alpha, x)$?

Intuition: The vector x satisfies the constraint α. So it should have no loss

The Identity Axiom

Consider an assignment to a set of Boolean variables (also called a *state*)

Let x denote its representation as a binary vector (to be interpreted as a probability)

Let α denote its representation as a logical sentence

What can we say about the loss $L(\alpha, x)$?

Intuition: The vector x satisfies the constraint α. So it should have no loss

The Identity axiom: *For any state x, there is zero semantic loss between its representation as a sentence and its representation as a deterministic vector.* $\forall x, L(x, x) = 0$.

What are some desirable properties of the loss function $L(\alpha, \mathbf{p})$?

1. The loss should be sub-differentiable

2. For two sentences α and β, if $\alpha \vDash \beta$, we want $L(\alpha, \mathbf{p}) \geq L(\beta, \mathbf{p})$
   → Logically equivalent sentences should have equal losses

3. There should be zero loss between a conjunction of literals and distribution corresponding to the hard assignment of variables that makes the conjunction `true`

Xu, Jingyi, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. "A semantic loss function for deep learning with symbolic knowledge." In *ICML 2018*

Labeled data as literals

Recall that labeled a labeled example is a proposition

The statement: “Example x has label y” is same as the proposition $\text{Label}(x, y)$

Similarly, “Example x doesn’t have the label y” is the negation $\neg\text{Label}(x, y)$

Labeled data as literals

Recall that labeled a labeled example is a proposition

The statement: "Example x has label y" is same as the proposition $\mathrm{Label}(x, y)$

Similarly, "Example x doesn't have the label y" is the negation $\neg\mathrm{Label}(x, y)$

Suppose we have a model producing a probability p for such propositions

Labeled data as literals

Recall that labeled a labeled example is a proposition

The statement: "Example x has label y" is same as the proposition $\mathrm{Label}(x, y)$

Similarly, "Example x doesn't have the label y" is the negation $\neg\mathrm{Label}(x, y)$

Suppose we have a model producing a probability p for such propositions

What do we expect from the losses $\mathrm{L}(\mathrm{X}, \mathrm{p})$ and $\mathrm{L}(\neg\mathrm{X}, \mathrm{p})$?

The Label-Literal Correspondence Axiom

Recall that labeled a labeled example is a proposition

- The statement: "Example x has label y" is same as the proposition $\text{Label}(x, y)$
- Similarly, "Example x doesn't have the label y" is the negation $\neg\text{Label}(x, y)$

Suppose we have a model producing a probability p for such propositions

What do we expect from the losses $L(X, p)$ and $L(\neg X, p)$?

Label-literal correspondence axiom:

$$L(X, p) \propto -\log p$$
$$L(\neg X, p) \propto -\log(1 - p)$$

What are some desirable properties of the loss function $L(\alpha, \mathbf{p})$?

1. The loss should be sub-differentiable

2. For two sentences α and β, if $\alpha \vDash \beta$, we want $L(\alpha, \mathbf{p}) \geq L(\beta, \mathbf{p})$
   → Logically equivalent sentences should have equal losses

3. There should be zero loss between a conjunction of literals and distribution corresponding to the hard assignment of variables that makes the conjunction **`true`**

4. For a predicate corresponding to a labeled predicate, the loss should be negative log of the probability of the predicate

Xu, Jingyi, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. "A semantic loss function for deep learning with symbolic knowledge." In *ICML 2018*

A consequence of the label-literal correspondence axiom

If it is true that:

$$L(X, p) \propto -\log p$$
$$L(\neg X, p) \propto -\log(1 - p)$$

A consequence of the label-literal correspondence axiom

If it is true that:

$$L(X, p) \propto -\log p$$
$$L(\neg X, p) \propto -\log(1-p)$$

Then, for any state x (i.e. a conjunction of literals), we have

$$L(x, \mathbf{p}) \propto -\sum_{i:x \models X_i} \log p_i - \sum_{i:x \models \neg X_i} \log(1-p_i)$$

A consequence of the label-literal correspondence axiom

If it is true that:

$$L(X, p) \propto -\log p$$
$$L(\neg X, p) \propto -\log(1-p)$$

Then, for any state x (i.e. a conjunction of literals), we have

$$L(x, \mathbf{p}) \propto -\sum_{i:x \vDash X_i} \log p_i - \sum_{i:x \vDash \neg X_i} \log(1-p_i)$$

Let us see an example. Suppose $x = X_1 \wedge \neg X_2 \wedge X_3$

A consequence of the label-literal correspondence axiom

If it is true that:

$$L(X, p) \propto -\log p$$
$$L(\neg X, p) \propto -\log(1-p)$$

Then, for any state x (i.e. a conjunction of literals), we have

$$L(x, \mathbf{p}) \propto -\sum_{i:x \vDash X_i} \log p_i - \sum_{i:x \vDash \neg X_i} \log(1-p_i)$$

Let us see an example. Suppose $x = X_1 \wedge \neg X_2 \wedge X_3$

For any probability $\mathbf{p} = [p_1, p_2, p_3]$ over the three literals we have

$$L(x, \mathbf{p}) \propto -\log p_1 - \log p_3 - \log(1-p_2)$$

What are some desirable properties of the loss function $L(\alpha, \mathbf{p})$?

1. The loss should be sub-differentiable

2. For two sentences α and β, if $\alpha \models \beta$, we want $L(\alpha, \mathbf{p}) \geq L(\beta, \mathbf{p})$
   → Logically equivalent sentences should have equal losses

3. There should be zero loss between a conjunction of literals and distribution corresponding to the hard assignment of variables that makes the conjunction **`true`**

4. For a predicate corresponding to a labeled predicate, the loss should be negative log of the probability of the predicate
   → For a conjunction of literals, the loss is the negative sum of the probabilities of the literals (accounting for polarity appropriately)

Xu, Jingyi, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. "A semantic loss function for deep learning with symbolic knowledge." In *ICML 2018*

Logic as loss: Semantic loss

- Building up to semantic loss: The axioms
- Semantic loss
- Examples
  - Conjunction
  - Implication
- Complex constraints & Weighted Model Counting
  - Example: The exactly-one constraint

Semantic loss

The only function that satisfies these axioms up to a multiplicative constant is the *semantic loss*, defined as

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i : x \models X_i} p_i \cdot \prod_{i : x \models \neg X_i} (1 - p_i) \right)$$

Semantic loss

The only function that satisfies these axioms up to a multiplicative constant is the *semantic loss*, defined as

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i : x \models X_i} p_i \cdot \prod_{i : x \models \neg X_i} (1 - p_i) \right)$$

A statement in logic using variables $X_1, X_2, \cdots$

Semantic loss

The only function that satisfies these axioms up to a multiplicative constant is the *semantic loss*, defined as

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i : x \models X_i} p_i \cdot \prod_{i : x \models \neg X_i} (1 - p_i) \right)$$

A statement in logic using variables $X_1, X_2, \cdots$

A vector of probabilities for each of $X_1, X_2, \cdots$ being `true`, to be produced by some neural network

Semantic loss

The only function that satisfies these axioms up to a multiplicative constant is the *semantic loss*, defined as

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{\boxed{x \models \alpha}} \left(\prod_{i : x \models X_i} p_i \cdot \prod_{i : x \models \neg X_i} (1 - p_i) \right)$$

x is a model for the statement α (equivalently, an assignment to the variables that makes α `true`)

Semantic loss

The only function that satisfies these axioms up to a multiplicative constant is the *semantic loss*, defined as

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \vDash \alpha} \left(\prod_{i : x \vDash X_i} p_i \cdot \prod_{i : x \vDash \neg X_i} (1 - p_i) \right)$$

The variable X_i is `true` in x

x is a model for the statement α (equivalently, an assignment to the variables that makes α `true`)

Semantic loss

The only function that satisfies these axioms up to a multiplicative constant is the *semantic loss*, defined as

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i : x \models X_i} p_i \cdot \prod_{i : x \models \neg X_i} (1 - p_i) \right)$$

The variable X_i is `true` in x

The variable X_i is `false` in x

x is a model for the statement α (equivalently, an assignment to the variables that makes α `true`)

Semantic loss

The only function that satisfies these axioms up to a multiplicative constant is the *semantic loss*, defined as

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \vDash \alpha} \left(\prod_{i : x \vDash X_i} p_i \cdot \prod_{i : x \vDash \neg X_i} (1 - p_i) \right)$$

For all variables that are entailed by the model, the product of the probabilities that they are `true`

Semantic loss

The only function that satisfies these axioms up to a multiplicative constant is the *semantic loss*, defined as

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i : x \models X_i} p_i \cdot \prod_{i : x \models \neg X_i} (1 - p_i) \right)$$

For all variables that are entailed by the model, the product of the probabilities that they are `true`

For all variables that are contradicted by the model, the product of the probabilities that they are `false`

Semantic loss

The only function that satisfies these axioms up to a multiplicative constant is the *semantic loss*, defined as

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i: x \models X_i} p_i \cdot \prod_{i: x \models \neg X_i} (1 - p_i) \right)$$

The probability assigned by the model to this particular assignment of the variables

Semantic loss

The only function that satisfies these axioms up to a multiplicative constant is the *semantic loss*, defined as

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \vDash \alpha} \left(\prod_{i: x \vDash X_i} p_i \cdot \prod_{i: x \vDash \neg X_i} (1 - p_i) \right)$$

Sum over every possible model for the statement α. That is, sum over every assignment to the variables that makes α `true`.

The probability assigned by the model to this particular assignment of the variables

Semantic loss

The only function that satisfies these axioms up to a multiplicative constant is the *semantic loss*, defined as

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i : x \models X_i} p_i \cdot \prod_{i : x \models \neg X_i} (1 - p_i) \right)$$

Probability of generating some state (i.e. assignment) that satisfies the constraint α

Semantic loss

The only function that satisfies these axioms up to a multiplicative constant is the *semantic loss*, defined as

$$L(\alpha, \mathbf{p}) \propto \boxed{-\log \sum_{x \vDash \alpha} \left(\prod_{i : x \vDash X_i} p_i \cdot \prod_{i : x \vDash \neg X_i} (1 - p_i) \right)}$$

Semantic loss = Negative log probability of generating some state (i.e. assignment) that satisfies the constraint α

Logic as loss: Semantic loss

- Building up to semantic loss: The axioms

- Semantic loss

- Examples
  - Conjunction
  - Implication

- Complex constraints & Weighted Model Counting
  - Example: The exactly-one constraint

Example 1: A conjunction

Consider the conjunction $\alpha = X_1 \wedge X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i:x \models X_i} p_i \cdot \prod_{i:x \models \neg X_i} (1 - p_i) \right)$$

Example 1: A conjunction

Consider the conjunction $\alpha = X_1 \wedge X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i: x \models X_i} p_i \cdot \prod_{i: x \models \neg X_i} (1 - p_i) \right)$$

Example 1: A conjunction

Consider the conjunction $\alpha = X_1 \wedge X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

The conjunction has only one satisfying assignment: $(X_1 = \top, X_2 = \top)$

- The summation has only one element

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i : x \models X_i} p_i \cdot \prod_{i : x \models \neg X_i} (1 - p_i) \right)$$

Example 1: A conjunction

Consider the conjunction $\alpha = X_1 \wedge X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

The conjunction has only one satisfying assignment: $(X_1 = \top, X_2 = \top)$

- The summation has only one element

Both variables in the satisfying assignment are `true`.

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i: x \models X_i} p_i \cdot \prod_{i: x \models \neg X_i} (1 - p_i) \right)$$

Example 1: A conjunction

Consider the conjunction $\alpha = X_1 \wedge X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

The conjunction has only one satisfying assignment: $(X_1 = \top, X_2 = \top)$

- The summation has only one element

Both variables in the satisfying assignment are `true`.

$$L(\alpha, \mathbf{p}) \propto -\log p_1 p_2$$

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \vDash \alpha} \left(\prod_{i: x \vDash X_i} p_i \cdot \prod_{i: x \vDash \neg X_i} (1 - p_i) \right)$$

Example 1: A conjunction

Consider the conjunction $\alpha = X_1 \wedge X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

The conjunction has only one satisfying assignment: $(X_1 = \top, X_2 = \top)$

- The summation has only one element

Both variables in the satisfying assignment are `true`.

$$L(\alpha, \mathbf{p}) \propto -\log p_1 p_2$$

Let us examine the truth table for this formula

X_1	X_2	α
$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i: x \models X_i} p_i \cdot \prod_{i: x \models \neg X_i} (1 - p_i) \right)$$

Example 1: A conjunction

Consider the conjunction $\alpha = X_1 \wedge X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

The conjunction has only one satisfying assignment: $(X_1 = \top, X_2 = \top)$

The summation has only one element

X_1	X_2	α	probability
$\top$	$\top$	$\top$	$p_1 p_2$
$\top$	$\bot$	$\bot$	$p_1(1 - p_2)$
$\bot$	$\top$	$\bot$	$p_2(1 - p_1)$
$\bot$	$\bot$	$\bot$	$(1 - p_1)(1 - p_2)$

Both variables in the satisfying assignment are `true`.

$$L(\alpha, \mathbf{p}) \propto -\log p_1 p_2$$

Example 1: A conjunction

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i : x \models X_i} p_i \cdot \prod_{i : x \models \neg X_i} (1 - p_i) \right)$$

Consider the conjunction $\alpha = X_1 \wedge X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

The conjunction has only one satisfying assignment: $(X_1 = \top, X_2 = \top)$

The summation has only one element

Both variables in the satisfying assignment are `true`.

$$L(\alpha, \mathbf{p}) \propto -\log p_1 p_2$$

X_1	X_2	α	probability
$\top$	$\top$	$\top$	$p_1 p_2$
$\top$	$\bot$	$\bot$	$p_1(1 - p_2)$
$\bot$	$\top$	$\bot$	$p_2(1 - p_1)$
$\bot$	$\bot$	$\bot$	$(1 - p_1)(1 - p_2)$

Only this row contributes to the loss. Any probability allocated other rows is undesirable because they do not satisfy the formula.

Logic as loss: Semantic loss

- Building up to semantic loss: The axioms
- Semantic loss
- Examples
  - Conjunction
  - Implication
- Complex constraints & Weighted Model Counting
  - Example: The exactly-one constraint

Example 2: Implication

Consider the implication $\alpha = X_1 \rightarrow X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i:x \models X_i} p_i \cdot \prod_{i:x \models \neg X_i} (1 - p_i) \right)$$

Example 2: Implication

Consider the implication $\alpha = X_1 \rightarrow X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i:x \models X_i} p_i \cdot \prod_{i:x \models \neg X_i} (1 - p_i) \right)$$

Example 2: Implication

Consider the implication $\alpha = X_1 \rightarrow X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

The implication has three satisfying assignments (i.e. three models):

$(X_1 = \top, X_2 = \top)$ $(X_1 = \perp, X_2 = \top)$ $(X_1 = \perp, X_2 = \perp)$

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i: x \models X_i} p_i \cdot \prod_{i: x \models \neg X_i} (1 - p_i) \right)$$

Example 2: Implication

Consider the implication $\alpha = X_1 \rightarrow X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

The implication has three satisfying assignments (i.e. three models):

$$(X_1 = \top, X_2 = \top) \quad (X_1 = \bot, X_2 = \top) \quad (X_1 = \bot, X_2 = \bot)$$

$$\downarrow \quad \downarrow \quad \downarrow$$

$$X_1 \wedge X_2 \quad \neg X_1 \wedge X_2 \quad \neg X_1 \wedge \neg X_2$$

We can equivalently rewrite the three models for α as these terms

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\boxed{\prod_{i:x \models X_i} p_i \cdot \prod_{i:x \models \neg X_i} (1 - p_i)} \right)$$

Example 2: Implication

Consider the implication $\alpha = X_1 \to X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

The implication has three satisfying assignments (i.e. three models):

$$(X_1 = \top, X_2 = \top) \qquad (X_1 = \bot, X_2 = \top) \qquad (X_1 = \bot, X_2 = \bot)$$

$$\downarrow \qquad\qquad \downarrow \qquad\qquad \downarrow$$

$$X_1 \wedge X_2 \qquad \neg X_1 \wedge X_2 \qquad \neg X_1 \wedge \neg X_2$$

Let us examine each model separately and construct the innermost product

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i : x \models X_i} p_i \cdot \prod_{i : x \models \neg X_i} (1 - p_i) \right)$$

Example 2: Implication

Consider the implication $\alpha = X_1 \rightarrow X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

The implication has three satisfying assignments:

$X_1 \wedge X_2 \models X_1$ and $X_1 \wedge X_2 \models X_2$

$$X_1 \wedge X_2 \quad \neg X_1 \wedge X_2 \quad \neg X_1 \wedge \neg X_2$$

$$X_1 \wedge X_2 \longrightarrow p_1 p_2$$

Example 2: Implication

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i : x \models X_i} p_i \cdot \prod_{i : x \models \neg X_i} (1 - p_i) \right)$$

Consider the implication $\alpha = X_1 \rightarrow X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

The implication has three satisfying assignments:

$X_1 \wedge X_2 \rightarrow p_1 p_2$

$\neg X_1 \wedge X_2 \models \neg X_1$ and $\neg X_1 \wedge X_2 \models X_2$

$\neg X_1 \wedge X_2 \rightarrow (1 - p_1)\, p_2$

$\neg X_1 \wedge \neg X_2$

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i:x \models X_i} p_i \cdot \prod_{i:x \models \neg X_i} (1 - p_i) \right)$$

Example 2: Implication

Consider the implication $\alpha = X_1 \rightarrow X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

The implication has three satisfying assignments:

$X_1 \wedge X_2$	$\neg X_1 \wedge X_2$	$\neg X_1 \wedge \neg X_2$
$\downarrow$	$\downarrow$	$\downarrow$
$p_1 p_2$	$(1 - p_1)\, p_2$	$(1 - p_1)\,(1 - p_2)$

$\neg X_1 \wedge \neg X_2 \models \neg X_1$
and
$\neg X_1 \wedge \neg X_2 \models \neg X_2$

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i: x \models X_i} p_i \cdot \prod_{i: x \models \neg X_i} (1 - p_i) \right)$$

Example 2: Implication

Consider the implication $\alpha = X_1 \to X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

The implication has three satisfying assignments:

$X_1 \wedge X_2$ $\qquad$ $\neg X_1 \wedge X_2$ $\qquad$ $\neg X_1 \wedge \neg X_2$

$$p_1 p_2 + (1 - p_1)\, p_2 + (1 - p_1)\,(1 - p_2)$$

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i: x \models X_i} p_i \cdot \prod_{i: x \models \neg X_i} (1 - p_i) \right)$$

Example 2: Implication

Consider the implication $\alpha = X_1 \rightarrow X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

The implication has three satisfying assignments:

$X_1 \wedge X_2$ $\qquad$ $\neg X_1 \wedge X_2$ $\qquad$ $\neg X_1 \wedge \neg X_2$

$$p_1 p_2 + 1 - p_1$$

Example 2: Implication

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i:x \models X_i} p_i \cdot \prod_{i:x \models \neg X_i} (1 - p_i) \right)$$

Consider the implication $\alpha = X_1 \rightarrow X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

The implication has three satisfying assignments:

$X_1 \wedge X_2$ $\qquad$ $\neg X_1 \wedge X_2$ $\qquad$ $\neg X_1 \wedge \neg X_2$

$$L(\alpha, \mathbf{p}) \propto -\log(\, p_1 p_2 \; + \; 1 - p_1)$$

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i: x \models X_i} p_i \cdot \prod_{i: x \models \neg X_i} (1 - p_i) \right)$$

Example 2: Implication

Consider the implication $\alpha = X_1 \to X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

The implication has three satisfying assignments. Shown in the table here

$$L(\alpha, \mathbf{p}) \propto -\log(p_1 p_2 + 1 - p_1)$$

X_1	X_2	α
$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\bot$	$\top$	$\top$
$\bot$	$\bot$	$\top$

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i: x \models X_i} p_i \cdot \prod_{i: x \models \neg X_i} (1 - p_i) \right)$$

Example 2: Implication

Consider the implication $\alpha = X_1 \rightarrow X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

The implication has three satisfying assignments. Shown in the table here

X_1	X_2	α	probability
$\top$	$\top$	$\top$	$p_1 p_2$
$\top$	$\bot$	$\bot$	$p_1(1 - p_2)$
$\bot$	$\top$	$\top$	$p_2(1 - p_1)$
$\bot$	$\bot$	$\top$	$(1 - p_1)(1 - p_2)$

$$L(\alpha, \mathbf{p}) \propto -\log(p_1 p_2 \ + \ 1 - p_1)$$

$$L(\alpha, \mathbf{p}) \propto -\log \sum_{x \models \alpha} \left(\prod_{i: x \models X_i} p_i \cdot \prod_{i: x \models \neg X_i} (1 - p_i) \right)$$

Example 2: Implication

Consider the implication $\alpha = X_1 \rightarrow X_2$ over two variables

Suppose we have some neural network that produces probabilities p_1 and p_2 for X_1 and X_2 respectively being `true`

Let us work out the semantic loss $\mathrm{L}(\alpha, \mathbf{p})$

The implication has three satisfying assignments. Shown in the table here

X_1	X_2	α	probability
$\top$	$\top$	$\top$	$p_1 p_2$
$\top$	$\bot$	$\bot$	$p_1(1 - p_2)$
$\bot$	$\top$	$\top$	$p_2(1 - p_1)$
$\bot$	$\bot$	$\top$	$(1 - p_1)(1 - p_2)$

$$L(\alpha, \mathbf{p}) \propto -\log(p_1 p_2 \ + \ 1 - p_1)$$

These rows contribute to the loss. Any probability allocated the other row is undesirable because it does not satisfy the formula.

Summary: Semantic loss

An axiomatic approach for converting logic to loss functions

- Produces differentiable losses
- Equivalent to cross-entropy when we have labeled examples

Key technical component

- Sum over the probabilities of assignments that satisfy the Boolean expression
- In practice: compile to tractable representations, and if this produces a small enough expression, we can perform forward and backward passes using standard tools
- Other approaches possible. E.g. approximation

Pros and cons

- Well defined semantics, syntactic variations don't matter
- But, could hide a difficult computational problem in the innermost loop of gradient based optimization