Logic as Loss: T-norm losses

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

This lecture

- Quick refresher of propositional logic
- Multi-valued logics
- T-norms and their associated functions
- Logic to losses via t-norms
- Examples
  - Example 1: Conjunction
  - Example 2: Implication
- Syntax versus semantics
  - Example 3: The exactly one constraint

This lecture

- Quick refresher of propositional logic
- Multi-valued logics
- T-norms and their associated functions
- Logic to losses via t-norms
- Examples
  - Example 1: Conjunction
  - Example 2: Implication
- Syntax versus semantics
  - Example 3: The exactly one constraint

Let us revisit propositional logic

Propositional logic is a language comprising of

1. A set of constants: $\{\top, \bot\}$
   - One of these values is considered `true`

2. Propositional variables whose value can take any of the constant values

3. A set of connectives: $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$

Semantics of propositional logic

Formally defined via the semantics of its constants and the connectives

Key concept: *interpretation* = assignment to variables

Semantics of propositional logic

Formally defined via the semantics of its constants and the connectives

Key concept: *interpretation* = assignment to variables

Constants: Every interpretation entails $\top$, no interpretation entails $\bot$

Semantics of propositional logic

Formally defined via the semantics of its constants and the connectives

Key concept: *interpretation* = assignment to variables

Constants: Every interpretation entails $\top$, no interpretation entails $\bot$

Connectives: Each connective has its own semantics

(We have seen this before)

Example: for conjunctions

An interpretation I is a model for a formula $F_1 \wedge F_2$ if, and only if, the interpretation is a model for the formula F_1 and a model for the formula F_2

$$I \vDash F_1 \wedge F_2 \quad \text{iff} \quad I \vDash F_1 \text{ and } I \vDash F_2$$

Another way to define semantics: Logical matrices

(Basically a different way to write down truth tables)

	$\neg$
$\top$	$\bot$
$\bot$	$\top$

Negation

$\wedge$	$\top$	$\bot$
$\top$	$\top$	$\top$
$\bot$	$\top$	$\bot$

Conjunction

$\vee$	$\top$	$\bot$
$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$

Disjunction

$\rightarrow$	$\top$	$\bot$
$\top$	$\top$	$\bot$
$\bot$	$\top$	$\top$

Material implication

$\leftrightarrow$	$\top$	$\bot$
$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\top$

Biconditional

Another way to define semantics: Logical matrices

(Basically a different way to write down truth tables)

	$\neg$
$\top$	$\bot$
$\bot$	$\top$

Negation

$\wedge$	$\top$	$\bot$
$\top$	$\top$	$\top$
$\bot$	$\top$	$\bot$

Conjunction

$\vee$	$\top$	$\bot$
$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$

Disjunction

$\rightarrow$	$\top$	$\bot$
$\top$	$\top$	$\bot$
$\bot$	$\top$	$\top$

Material implication

$\leftrightarrow$	$\top$	$\bot$
$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\top$

Biconditional

$\top \wedge \bot$

Another way to define semantics: Logical matrices

(Basically a different way to write down truth tables)

	$\neg$
$\top$	$\bot$
$\bot$	$\top$

Negation

$\wedge$	$\top$	$\bot$
$\top$	$\top$	$\top$
$\bot$	$\top$	$\bot$

Conjunction

$\vee$	$\top$	$\bot$
$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$

Disjunction

$\rightarrow$	$\top$	$\bot$
$\top$	$\top$	$\bot$
$\bot$	$\top$	$\top$

Material implication

$\leftrightarrow$	$\top$	$\bot$
$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\top$

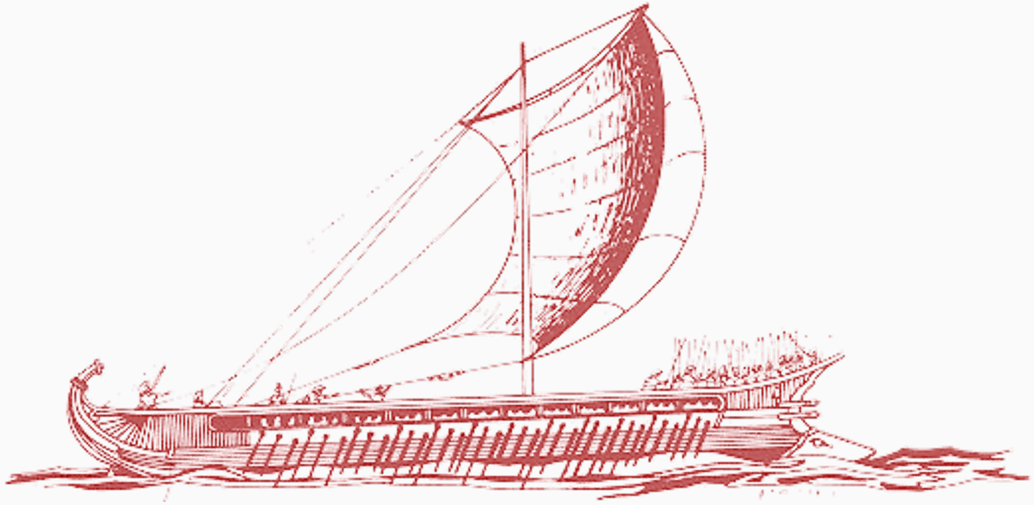
Biconditional

$\top \wedge \bot = \bot$

This lecture

- Quick refresher of propositional logic
- Multi-valued logics
- T-norms and their associated functions
- Logic to losses via t-norms
- Examples
  - Example 1: Conjunction
  - Example 2: Implication
- Syntax versus semantics
  - Example 3: The exactly one constraint

The sea battle problem



Consider the following two sentences:

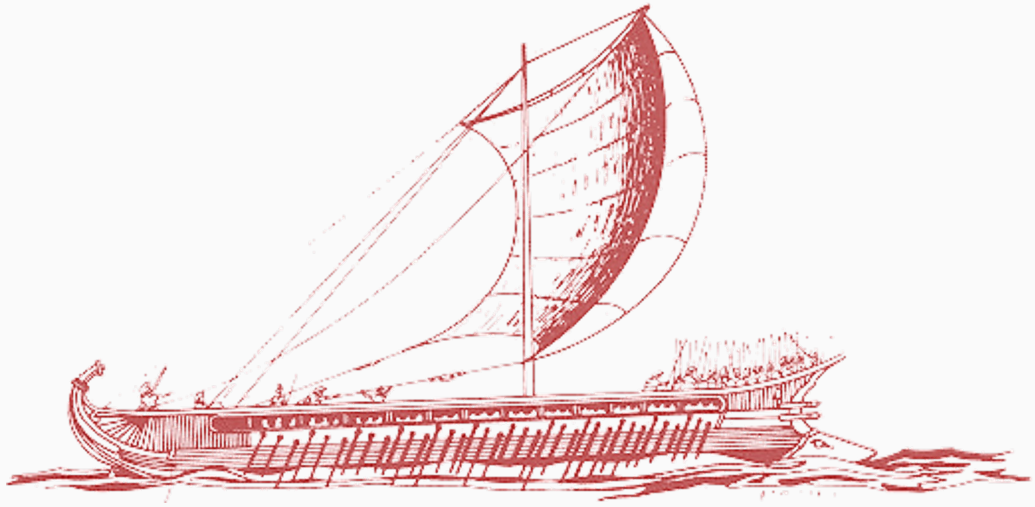
A: There will be a sea battle tomorrow

B: There will not be a sea battle tomorrow

Clearly A and B cannot both be true

But how can we assign a truth value to either one of them *today*?

The sea battle problem



Consider the following two sentences:

A: There will be a sea battle tomorrow

B: There will not be a sea battle tomorrow

Clearly A and B cannot both be true

But how can we assign a truth value to either one of them *today*?

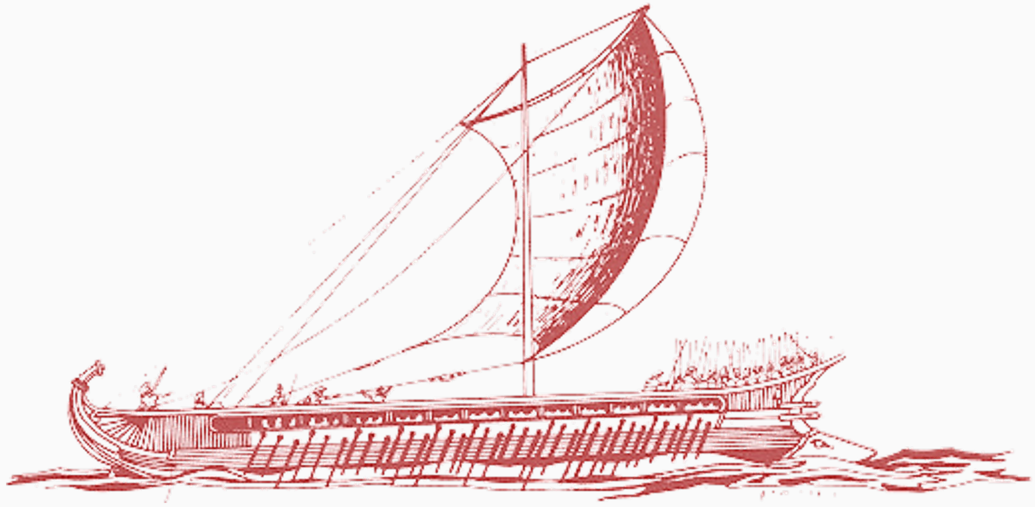
The same problem doesn't apply to past or present events

A': There was a sea battle yesterday

B': There was no sea battle yesterday

Clearly only one of these is true

The sea battle problem



Consider the following two sentences:

A: There will be a sea battle tomorrow

B: There will not be a sea battle tomorrow

Clearly A and B cannot both be true

But how can we assign a truth value to either one of them *today*?

This is also called the problem of *future contingents*

Statements about future events, actions, etc that are neither impossible nor inevitable

This example and the problem goes back to Aristotle's "On Interpretation" (Chapter 9)

Several attempts over the centuries to address this within propositional logic

An attempt at a solution

The problem arises because propositional logic does not have a place for ideas like "maybe true", "maybe false", "don't know", etc.

Jan Łukasiewicz (1920): Let's add a third constant denoting "possible" and extend the meaning of the standard operators to define

Łukasiewicz's three-valued logic

Łukasiewicz's three-valued logic: $Ł_3$

A language comprising of:

1. A set of constants: $\{0, ½, 1\}$
   - Only the constant 1 is designated as `true`
   - The constant ½ could be interpreted as "possible" or "as of now undetermined"
   - The constant 0 is `false`

2. Propositional variables whose value can take any of the constant values

3. A set of connectives: $\{\neg, \rightarrow\}$

Semantics of the connectors in $Ł_3$

We can describe the connectives using the logic matrices as before.

This time, there will be three possible inputs for each operand

Semantics of the connectors in $Ł_3$

	$\neg$
1	0
½	½
0	1

Semantics of the connectors in $Ł_3$

	$\neg$
1	0
½	½
0	1

$\neg x = 1 - x$

Semantics of the connectors in $Ł_3$

	$\neg$
1	0
½	½
0	1

$\rightarrow$	1	½	0
1	1	½	0
½	1	1	½
0	1	1	1

$\neg x = 1 - x$

Semantics of the connectors in $Ł_3$

	$\neg$
1	0
½	½
0	1

$\neg x = 1 - x$

$\to$	1	½	0
1	1	½	0
½	1	1	½
0	1	1	1

$x \to y = \min(1, 1 - x + y)$

Semantics of the connectors in Ł$_3$

	$\neg$
1	0
½	½
0	1

$\rightarrow$	1	½	0
1	1	½	0
½	1	1	½
0	1	1	1

$\neg x = 1 - x$

$x \rightarrow y = \min(1, 1 - x + y)$

What do these connectives mean?

Semantics of the connectors in Ł$_3$

	$\neg$
1	0
½	½
0	1

$\rightarrow$	1	½	0
1	1	½	0
½	1	1	½
0	1	1	1

$\neg x = 1 - x$

$x \rightarrow y = \min(1, 1 - x + y)$

What do these connectives mean?

There are other three-valued logics, which have their own set of connectives

There are also four-valued logics, nine-value logics and finite-valued logics

Łukasiewicz logic: An infinite valued logic

A language Ł comprising of:

1. An infinite set of constants: $[0, 1]$
   - Only the value 1 is designated as `true`
   - The value 0 is `false`
   - The values in between reflect degrees of truth

2. Propositional variables whose value can take any of the constant values

3. A set of connectives: $\{\rightarrow\}$

Łukasiewicz logic: An infinite valued logic

A language Ł comprising of:

1. An infinite set of constants: $[0, 1]$
   - Only the value 1 is designated as `true`
   - The value 0 is `false`
   - The values in between reflect degrees of truth

2. Propositional variables whose value can take any of the constant values

3. A set of connectives: $\{\rightarrow\}$

Degree of truth

In classical logic, all statements are unequivocally true or false

Is this always the case?

Can we have statements in real life that are not unambiguously true or false?

Let’s brainstorm

Degree of truth

In classical logic, all statements are unequivocally true or false

What about sentences like:

- Salt Lake City is a large city
- Kalamazoo is a large city

- Bach’s Concerto No. 1 is a beautiful piece of music
- Iron Maiden’s Fear of the Dark is a beautiful piece of music

- Aristotle was a heavy baby when he was born
- Aristotle was a small baby when he was born

Degree of truth

In classical logic, all statements are unequivocally true or false

What about sentences like:

- Salt Lake City is a large city
- Kalamazoo is a large city

- Bach’s Concerto No. 1 is a beautiful piece of music
- Iron Maiden’s Fear of the Dark is a beautiful piece of music

- Aristotle was a heavy baby when he was born
- Aristotle was a small baby when he was born

In each case, the answer is subjective.

The statements are true to some degree

Degree of truth

In classical logic, all statements are unequivocally true or false

What about sentences like:

- Salt Lake City is a large city
- Kalamazoo is a large city

- Bach's Concerto No. 1 is a beautiful piece of music
- Iron Maiden's Fear of the Dark is a beautiful piece of music

- Aristotle was a heavy baby when he was born
- Aristotle was a small baby when he was born

In each case, the answer is subjective.

The statements are true to some degree

Degree of truth is not the same as degree of belief

Degree of belief describes the probability that some statement is true or not

Degree of truth describes the vagueness inherent in the truth value of a statement

Łukasiewicz logic: An infinite valued logic

A language Ł comprising of:

1. An infinite set of constants: $[0, 1]$
   - Only the value 1 is designated as `true`
   - The value 0 is `false`
   - The values in between reflect degrees of truth

2. Propositional variables whose value can take any of the constant values

3. A set of connectives: $\{\rightarrow\}$

Łukasiewicz logic: An infinite valued logic

A language Ł comprising of:

1. An infinite set of constants: $[0, 1]$
   - Only the value 1 is designated as `true`
   - The value 0 is `false`
   - The values in between reflect degrees of truth

2. Propositional variables whose value can take any of the constant values

3. A set of connectives: $\{\rightarrow\}$ Let's see the definition of this connective next

Semantics of the connectors in Ł

Same as the definition of implication in the three-valued logic $Ł_3$

We can't write a logic matrice because the connectives operate on an infinite set.
Instead, we write the definition as function:

$$x \to y = \min(1, 1 - x + y)$$

Semantics of the connectors in Ł

Same as the definition of implication in the three-valued logic $Ł_3$

We can't write a logic matrice because the connectives operate on an infinite set. Instead, we write the definition as function:

$$x \rightarrow y = \min(1, 1 - x + y)$$

But what about the other connectives?

Semantics of the connectors in Ł

Same as the definition of implication in the three-valued logic $Ł_3$

We can't write a logic matrice because the connectives operate on an infinite set.
Instead, we write the definition as function:

$$x \rightarrow y = \min(1, 1 - x + y)$$

But what about the other connectives?

They can all be defined in terms of the implication

Other connectives from the implication $\to$

$\neg x$ is equivalent to $x \to \bot$

$x \vee y$ is equivalent to $\neg x \to y$

$x \wedge y$ is equivalent to $\neg(\neg x \vee \neg y)$ by De Morgan's law

Easy to verify all of these are true for the propositional case

We can write truth tables

Other connectives from the implication $\to$

$\neg x$ is equivalent to $x \to \bot$

$x \lor y$ is equivalent to $\neg x \to y$

$x \land y$ is equivalent to $\neg(\neg x \lor \neg y)$ by De Morgan's law

Easy to verify all of these are true for the propositional case

We can write truth tables

Side note: There are other ways to define the disjunction too. What we have here is called the strong disjunction in some places, and this wasn't in Łukasiewicz's original work

Other connectives from the implication $\rightarrow$

Łukasiewicz implication: $x \rightarrow y = \min(1, 1 - x + y)$

$\neg x$ is equivalent to $x \rightarrow \perp$

We can work out the full set of operators using these definitions

$x \vee y$ is equivalent to $\neg x \rightarrow y$

$x \wedge y$ is equivalent to $\neg(\neg x \vee \neg y)$ by De Morgan's law

Other connectives from the implication $\rightarrow$

Łukasiewicz implication: $x \rightarrow y = \min(1, 1 - x + y)$

$\neg x$ is equivalent to $x \rightarrow \perp$

$$\min(1, 1 - x + 0) = \min(1,1 - x) = 1 - x$$

$x \vee y$ is equivalent to $\neg x \rightarrow y$

$x \wedge y$ is equivalent to $\neg(\neg x \vee \neg y)$ by De Morgan's law

Other connectives from the implication $\rightarrow$

Łukasiewicz implication: $x \rightarrow y = \min(1, 1 - x + y)$

$\neg x$ is equivalent to $x \rightarrow \perp$

$$\min(1, 1 - x + 0) = \min(1,1 - x) = 1 - x$$

$x \vee y$ is equivalent to $\neg x \rightarrow y$

$$\min(1, 1 - (1 - x) + y) = \min(1, x + y)$$

$x \wedge y$ is equivalent to $\neg(\neg x \vee \neg y)$ by De Morgan’s law

Other connectives from the implication $\rightarrow$

Łukasiewicz implication: $x \rightarrow y = \min(1, 1 - x + y)$

$\neg x$ is equivalent to $x \rightarrow \perp$

$$\min(1, 1 - x + 0) = \min(1,1 - x) = 1 - x$$

$x \vee y$ is equivalent to $\neg x \rightarrow y$

$$\min(1, 1 - (1 - x) + y) = \min(1, x + y)$$

$x \wedge y$ is equivalent to $\neg(\neg x \vee \neg y)$ by De Morgan's law

$$1 - \min(1, 1 - x + 1 - y) = \max(0, x + y - 1)$$

Łukasiewicz logic

Atoms are numbers between 0 and 1, representing degrees of truth, or variables

Connectives defined as

Negation	$\neg x$	$1 - x$
Conjunction	$x \wedge y$	$\max(0, x + y - 1)$
Disjunction	$x \vee y$	$\min(1, x + y)$
Implication	$x \rightarrow y$	$\min(1, 1 - x + y)$

This logic is a specific instance of t-norm logics or fuzzy logics

This lecture

- Quick refresher of propositional logic
- Multi-valued logics
- T-norms and their associated functions
- Logic to losses via t-norms
- Examples
  - Example 1: Conjunction
  - Example 2: Implication
- Syntax versus semantics
  - Example 3: The exactly one constraint

Triangular norms (t-norms)

A t-norm is a generalization of a conjunction to the infinite valued case

Any function $T: [0,1]\times[0,1] \to [0,1]$ that satisfies the following properties is a t-norm

As we see the properties, check if they hold for conjunctions

Triangular norms (t-norms)

A t-norm is a generalization of a conjunction to the infinite valued case

Any function $T: [0,1]\times[0,1] \rightarrow [0,1]$ that satisfies the following properties is a t-norm

1. Commutativity: $T(x, y) = T(y, x)$

Triangular norms (t-norms)

A t-norm is a generalization of a conjunction to the infinite valued case

Any function $T\colon [0,1]\times[0,1] \rightarrow [0,1]$ that satisfies the following properties is a t-norm

1. Commutativity: $T(x, y) = T(y, x)$

2. Associativity: $T\big(x, T(y, z)\big) = T(T(x, y), z)$

Triangular norms (t-norms)

A t-norm is a generalization of a conjunction to the infinite valued case

Any function $T: [0,1] \times [0,1] \rightarrow [0,1]$ that satisfies the following properties is a t-norm

1. Commutativity: $T(x, y) = T(y, x)$
2. Associativity: $T\big(x, T(y, z)\big) = T(T(x, y), z)$
3. Monotonicity: For any x, y such that $x \leq y$, we have $T(x, z) \leq T(y, z)$

Triangular norms (t-norms)

A t-norm is a generalization of a conjunction to the infinite valued case

Any function $T: [0,1] \times [0,1] \rightarrow [0,1]$ that satisfies the following properties is a t-norm

1. Commutativity: $T(x, y) = T(y, x)$
2. Associativity: $T\big(x, T(y, z)\big) = T(T(x, y), z)$
3. Monotonicity: For any x, y such that $x \leq y$, we have $T(x, z) \leq T(y, z)$
4. Identity: $T(x, 1) = x$

Triangular norms (t-norms)

A t-norm is a generalization of a conjunction to the infinite valued case

Any function $T: [0,1]\times[0,1] \to [0,1]$ that satisfies the following properties is a t-norm

1. Commutativity: $T(x, y) = T(y, x)$
2. Associativity: $T\big(x, T(y, z)\big) = T(T(x, y), z)$
3. Monotonicity: For any x, y such that $x \leq y$, we have $T(x, z) \leq T(y, z)$
4. Identity: $T(x, 1) = x$

There are an infinite number of functions that satisfy these properties

Triangular norms (t-norms)

A t-norm is a generalization of a conjunction to the infinite valued case

Any function $T: [0,1]\times[0,1] \rightarrow [0,1]$ that satisfies the following properties is a t-norm

1. Commutativity: $T(x, y) = T(y, x)$
2. Associativity: $T\big(x, T(y, z)\big) = T(T(x, y), z)$
3. Monotonicity: For any x, y such that $x \leq y$, we have $T(x, z) \leq T(y, z)$
4. Identity: $T(x, 1) = x$

There are an infinite number of functions that satisfy these properties

Triangular norms (t-norms)

A t-norm is a generalization of a conjunction to the infinite valued case

Any function $T: [0,1]\times[0,1] \rightarrow [0,1]$ that satisfies the following properties is a t-norm

1. Commutativity: $T(x, y) = T(y, x)$
2. Associativity: $T\big(x, T(y, z)\big) = T(T(x, y), z)$
3. Monotonicity: For any x, y such that $x \leq y$, we have $T(x, z) \leq T(y, z)$
4. Identity: $T(x, 1) = x$

There are an infinite number of functions that satisfy these properties

We have already seen one of them:
The Łukasiewicz conjunction

Fundamental t-norms

Three t-norms are called the fundamental t-norms

1. Łukasiewicz t-norm: $\wedge_L\,(x, y) = \max(0, x + y - 1)$
2. Gödel t-norm: $\wedge_G\,(x, y) = \min(x, y)$
3. Product t-norm: $\wedge_P\,(x, y) = xy$

Conjunction : t-norm :: Disjunction : t-conorm

Like t-norms, we have t-conorms which generalize disjunctions to the infinite valued case

Any function $C: [0,1] \times [0,1] \rightarrow [0,1]$ that satisfies the following properties is a t-norm

As we see the properties, check if they hold for disjunctions

Conjunction : t-norm :: Disjunction : t-conorm

Like t-norms, we have t-conorms which generalize disjunctions to the infinite valued case

Any function $C: [0,1] \times [0,1] \rightarrow [0,1]$ that satisfies the following properties is a t-norm

1. Commutativity: $C(x, y) = C(y, x)$

Conjunction : t-norm :: Disjunction : t-conorm

Like t-norms, we have t-conorms which generalize disjunctions to the infinite valued case

Any function C: $[0,1]\times[0,1] \rightarrow [0,1]$ that satisfies the following properties is a t-norm

1. Commutativity: $C(x, y) = C(y, x)$

2. Associativity: $C\big(x, C(y, z)\big) = C(C(x, y), z)$

Conjunction : t-norm :: Disjunction : t-conorm

Like t-norms, we have t-conorms which generalize disjunctions to the infinite valued case

Any function $C: [0,1] \times [0,1] \to [0,1]$ that satisfies the following properties is a t-norm

1. Commutativity: $C(x, y) = C(y, x)$

2. Associativity: $C\big(x, C(y, z)\big) = C(C(x, y), z)$

3. Monotonicity: For any x, y such that $x \leq y$, we have $T(x, z) \leq T(y, z)$

Conjunction : t-norm :: Disjunction : t-conorm

Like t-norms, we have t-conorms which generalize disjunctions to the infinite valued case

Any function $C: [0,1]\times[0,1] \rightarrow [0,1]$ that satisfies the following properties is a t-norm

1. Commutativity: $C(x, y) = C(y, x)$
2. Associativity: $C\big(x, C(y, z)\big) = C(C(x, y), z)$
3. Monotonicity: For any x, y such that $x \leq y$, we have $T(x, z) \leq T(y, z)$
4. Identity: $T(x, 0) = x$

Conjunction : t-norm :: Disjunction : t-conorm

Like t-norms, we have t-conorms which generalize disjunctions to the infinite valued case

Any function $C: [0,1]\times[0,1] \rightarrow [0,1]$ that satisfies the following properties is a t-norm

1. Commutativity: $C(x, y) = C(y, x)$
2. Associativity: $C\big(x, C(y, z)\big) = C(C(x, y), z)$
3. Monotonicity: For any x, y such that $x \leq y$, we have $T(x, z) \leq T(y, z)$
4. Identity: $T(x, 0) = x$

There are an infinite number of functions that satisfy these properties

We have already seen one of them:
The Łukasiewicz disjunction

Every t-norm has an associated t-conorm

$$C(x, y) = 1 - T(1 - x, 1 - y)$$

This behaves like the De Morgan's rule for the t-norm

(assuming that the negation $\neg x$ is equivalent to $1 - x$)

This gives us the three fundamental t-conforms

Every t-norm has an associated t-conorm

$$C(x, y) = 1 - T(1 - x, 1 - y)$$

This behaves like the De Morgan's rule for the t-norm

(assuming that the negation $\neg x$ is equivalent to $1 - x$)

This gives us the three fundamental t-conforms

1. Łukasiewicz t-conorm: $\vee_L (x, y) = \min(1, x + y)$

Every t-norm has an associated t-conorm

$$C(x, y) = 1 - T(1 - x, 1 - y)$$

This behaves like the De Morgan's rule for the t-norm

(assuming that the negation $\neg x$ is equivalent to $1 - x$)

This gives us the three fundamental t-conforms

1. Łukasiewicz t-conorm: $\vee_L\ (x, y) = \min(1, x + y)$

2. Gödel t-conorm: $\vee_G\ (x, y) = \max(x, y)$

Every t-norm has an associated t-conorm

$$C(x, y) = 1 - T(1 - x, 1 - y)$$

This behaves like the De Morgan's rule for the t-norm

(assuming that the negation $\neg x$ is equivalent to $1 - x$)

This gives us the three fundamental t-conforms

1. Łukasiewicz t-conorm: $\vee_L\ (x, y) = \min(1, x + y)$

2. Gödel t-conorm: $\vee_G\ (x, y) = \max(x, y)$

3. Product t-conorm: $\vee_P\ (x, y) = x + y - xy$

Implications can be complicated

There are many different ways to define the → operation for a t-norm

Implications can be complicated

There are many different ways to define the $\rightarrow$ operation for a t-norm

One approach: The S-implication

In Boolean logic, we have $x \rightarrow y \equiv \neg x \vee y$

Implications can be complicated

There are many different ways to define the $\rightarrow$ operation for a t-norm

One approach: The S-implication

In Boolean logic, we have $x \rightarrow y \equiv \neg x \lor y$

Given a t-norm T, we know that its t-conorm C that represents the disjunction

Implications can be complicated

There are many different ways to define the $\rightarrow$ operation for a t-norm

One approach: The S-implication

In Boolean logic, we have $x \rightarrow y \equiv \neg x \vee y$

Given a t-norm T, we know that its t-conorm C that represents the disjunction

We generally assume that the negation $\neg x$ is $1 - x$

Implications can be complicated

There are many different ways to define the $\rightarrow$ operation for a t-norm

One approach: The S-implication

In Boolean logic, we have $x \rightarrow y \equiv \neg x \vee y$

Given a t-norm T, we know that its t-conorm C that represents the disjunction

We generally assume that the negation $\neg x$ is $1 - x$

This gives us a definition of the implication:

$$x \rightarrow_S y = C(1 - x, y)$$

Implications can be complicated

There are many different ways to define the → operation for a t-norm

Implications can be complicated

There are many different ways to define the → operation for a t-norm

Another approach: The R-implication, which calls the implication a residuum

(There is a formal definition, but that is too much of a distraction for now)

Implications can be complicated

There are many different ways to define the $\rightarrow$ operation for a t-norm

Another approach: The R-implication, which calls the implication a residuum

Captures the following intuition:

Given a t-norm that defines the conjunction, we want $x \wedge (x \rightarrow y)$ to be less than y

(There is a formal definition, but that is too much of a distraction for now)

Implications can be complicated

There are many different ways to define the $\rightarrow$ operation for a t-norm

Another approach: The R-implication, which calls the implication a residuum

Captures the following intuition:

Given a t-norm that defines the conjunction, we want $x \wedge (x \rightarrow y)$ to be less than y

There may be many functions that satisfy this criterion. Pick the one that makes the value of $x \rightarrow y$ maximum

(There is a formal definition, but that is too much of a distraction for now)

From Boolean logic to t-norm logics

Triangular norms provide systematic generalizations of logic
Some are continuous and sub-differentiable

	Boolean logic
Not	$\neg A$
And	$A \wedge B$
Or	$A \vee B$
Implies	$A \rightarrow B$

Klement, Erich Peter, Radko Mesiar, and Endre Pap. *Triangular norms*. Vol. 8. 2013.

From Boolean logic to t-norm logics

Triangular norms provide systematic generalizations of logic

Some are continuous and sub-differentiable

	Inputs, outputs live in {0,1}
	Boolean logic
Not	$\neg A$
And	$A \wedge B$
Or	$A \vee B$
Implies	$A \rightarrow B$

Klement, Erich Peter, Radko Mesiar, and Endre Pap. *Triangular norms*. Vol. 8. 2013.

From Boolean logic to t-norm logics

Triangular norms provide systematic generalizations of logic

Some are continuous and sub-differentiable

Inputs, outputs live in {0,1}

	Boolean logic	Product	Gödel	Łukasiewicz
Not	$\neg A$	$1-a$	$1-a$	$1-a$
And	$A \wedge B$	ab	$\min(a,b)$	$\max(0, a+b-1)$
Or	$A \vee B$	$a+b-ab$	$\max(a,b)$	$\min(1, a+b)$
Implies	$A \rightarrow B$	$\min\left(1, \frac{b}{a}\right)$	$\begin{cases} 1 & \text{if } b > a \\ b & \text{else} \end{cases}$	$\min(1, 1-a+b)$

Klement, Erich Peter, Radko Mesiar, and Endre Pap. *Triangular norms*. Vol. 8. 2013.

From Boolean logic to t-norm logics

Triangular norms provide systematic generalizations of logic

Some are continuous and sub-differentiable

	Boolean logic (Inputs, outputs live in {0,1})	Product (Inputs, outputs live in [0,1])	Gödel	Łukasiewicz
Not	$\neg A$	$1-a$	$1-a$	$1-a$
And	$A \wedge B$	ab	$\min(a,b)$	$\max(0, a+b-1)$
Or	$A \vee B$	$a+b-ab$	$\max(a,b)$	$\min(1, a+b)$
Implies	$A \rightarrow B$	$\min\left(1, \frac{b}{a}\right)$	$\begin{cases} 1 & \text{if } b > a \\ b & \text{else} \end{cases}$	$\min(1, 1-a+b)$

Klement, Erich Peter, Radko Mesiar, and Endre Pap. *Triangular norms*. Vol. 8. 2013.

From Boolean logic to t-norm logics

Triangular norms provide systematic generalizations of logic

Some are continuous and sub-differentiable

	Boolean logic (Inputs, outputs live in {0,1})	Product (Inputs, outputs live in [0,1])	Gödel	Łukasiewicz
Not	$\neg A$	$1-a$	$1-a$	$1-a$
And	$A \wedge B$	ab	$\min(a,b)$	$\max(0, a+b-1)$
Or	$A \vee B$	$a+b-ab$	$\max(a,b)$	$\min(1, a+b)$
Implies	$A \rightarrow B$	$\min\left(1, \frac{b}{a}\right)$	$\begin{cases} 1 & \text{if } b > a \\ b & \text{else} \end{cases}$	$\min(1, 1-a+b)$

Importantly, all these operators agree with the Boolean operators at the boundaries

Klement, Erich Peter, Radko Mesiar, and Endre Pap. *Triangular norms*. Vol. 8. 2013.

This lecture

- Quick refresher of propositional logic
- Multi-valued logics
- T-norms and their associated functions
- Logic to losses via t-norms
- Examples
  - Example 1: Conjunction
  - Example 2: Implication
- Syntax versus semantics
  - Example 3: The exactly one constraint

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)

$\downarrow$

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

Instead of treating models as functions that produce probabilities, treat them as functions that produce a *degree of truth* associated with the atomic statements

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

Instead of treating models as functions that produce probabilities, treat them as functions that produce a *degree of truth* associated with the atomic statements

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~~ degree of truth p_i, possibly from a neural model

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

Our goal:

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

The recipe for converting a formula into a loss

Given:

- A statement α composed of atoms $X_1, X_2, \cdots, X_n$
- Model predictions $p_1, p_2, \cdots, p_n$ representing the degree of truth for each atom
- A choice of a t-norm (and its associated functions)

The recipe for converting a formula into a loss

Given:

- A statement α composed of atoms $X_1, X_2, \cdots, X_n$
- Model predictions $p_1, p_2, \cdots, p_n$ representing the degree of truth for each atom
- A choice of a t-norm (and its associated functions)

Step 1: Replace every Boolean operator with its relaxation

The recipe for converting a formula into a loss

Given:

- A statement α composed of atoms $X_1, X_2, \cdots, X_n$
- Model predictions $p_1, p_2, \cdots, p_n$ representing the degree of truth for each atom
- A choice of a t-norm (and its associated functions)

Step 1: Replace every Boolean operator with its relaxation

Step 2: Find models maximize the relaxation

Or equivalently: Find models that minimize the negative log relaxation

The recipe for converting a formula into a loss

Given:

- A statement α composed of atoms $X_1, X_2, \cdots, X_n$
- Model predictions $p_1, p_2, \cdots, p_n$ representing the degree of truth for each atom
- A choice of a t-norm (and its associated functions)

Step 1: Replace every Boolean operator with its relaxation

Step 2: Find models maximize the relaxation

The loss function

Or equivalently: Find models that minimize the negative log relaxation

This lecture

- Quick refresher of propositional logic
- Multi-valued logics
- T-norms and their associated functions
- Logic to losses via t-norms
- Examples
  - Example 1: Conjunction
  - Example 2: Implication
- Syntax versus semantics
  - Example 3: The exactly one constraint

Example 1: A conjunction

Consider the conjunction $\alpha = X_1 \land X_2$ over two variables
Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α ($\text{written as } [\alpha]$).

Example 1: A conjunction

Consider the conjunction $\alpha = X_1 \wedge X_2$ over two variables
Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α ($\text{written as } [\alpha]$).

First, we need to pick a t-norm.

Example 1: A conjunction

Consider the conjunction $\alpha = X_1 \wedge X_2$ over two variables
Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α (written as $[\alpha]$).

First, we need to pick a t-norm. Let's choose Product t-norm with $a \wedge b = ab$

$$[\alpha] = [X_1 \wedge X_2] = p_1 p_2$$

Example 1: A conjunction

Consider the conjunction $\alpha = X_1 \land X_2$ over two variables
Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α (written as $[\alpha]$).

First, we need to pick a t-norm. Let's choose Product t-norm with $a \land b = ab$

$$[\alpha] = [X_1 \land X_2] = p_1 p_2$$

Our goal: To find models that make the statement α true.

Example 1: A conjunction

Consider the conjunction $\alpha = X_1 \land X_2$ over two variables
Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α (written as $[\alpha]$).

First, we need to pick a t-norm. Let's choose Product t-norm with $a \land b = ab$

$$[\alpha] = [X_1 \land X_2] = p_1 p_2$$

Our goal: To find models that make the statement α true.
Or equivalently: to find models that make the –ve log relaxation as low as possible

Example 1: A conjunction

Consider the conjunction $\alpha = X_1 \wedge X_2$ over two variables
Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α (written as $[\alpha]$).

First, we need to pick a t-norm. Let's choose Product t-norm with $a \wedge b = ab$

$$[\alpha] = [X_1 \wedge X_2] = p_1 p_2$$

Our goal: To find models that make the statement α true.
Or equivalently: to find models that make the –ve log relaxation as low as possible

$$L(\alpha, p) = -\log p_1 p_2 = -\sum_{i=1}^{2} \log p_i$$

A conjunction with more variables

Suppose we have a set of labeled examples: $\{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$

Recall: The labeled example (x_i, y_i) is a predicate that says

"Instance x_i has this label y_i" $= \text{Label}(x_i, y_i)$

A conjunction with more variables

Suppose we have a set of labeled examples: $\{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$

Recall: The labeled example (x_i, y_i) is a predicate that says

“Instance x_i has this label y_i” $=$ $\mathrm{Label}(x_i, y_i)$

Our labeled dataset is a large conjunction

$$\bigwedge_{i=1}^{m} \mathrm{Label}(x_i, y_i)$$

A conjunction with more variables

Suppose we have a set of labeled examples: $\{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$

Recall: The labeled example (x_i, y_i) is a predicate that says

"Instance x_i has this label y_i" $=$ $\text{Label}(x_i, y_i)$

Our labeled dataset is a large conjunction

$$\bigwedge_{i=1}^{m} \text{Label}(x_i, y_i)$$

The goal of learning is to make this conjunction true. Or to maximize its relaxation

A conjunction with more variables

Suppose we have a set of labeled examples: $\{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$

Recall: The labeled example (x_i, y_i) is a predicate that says

“Instance x_i has this label y_i” $= \text{Label}(x_i, y_i)$

Our labeled dataset is a large conjunction

$$\bigwedge_{i=1}^{m} \text{Label}(x_i, y_i)$$

The goal of learning is to make this conjunction true. Or to maximize its relaxation

Or to minimize its negative log relaxation. Using the product t-norm gives us a loss

A conjunction with more variables

Suppose we have a set of labeled examples: $\{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$

Recall: The labeled example (x_i, y_i) is a predicate that says

“Instance x_i has this label y_i” $= \text{Label}(x_i, y_i)$

Our labeled dataset is a large conjunction

$$\bigwedge_{i=1}^{m} \text{Label}(x_i, y_i)$$

The goal of learning is to make this conjunction true. Or to maximize its relaxation

Or to minimize its negative log relaxation. Using the product t-norm gives us a loss

$$-\log\left[\bigwedge_{i=1}^{m} \text{Label}(x_i, y_i)\right] = -\log \prod_{i=1}^{m} \text{model}(x_i, y_i) = -\sum_{i} \log \text{model}(x_i, y_i)$$

The degree of truth assigned by the model to label y_i for input x_i

A conjunction with more variables

Suppose we have a set of labeled examples: $\{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$

Recall: The labeled example (x_i, y_i) is a predicate that says

"Instance x_i has this label y_i" $= \text{Label}(x_i, y_i)$

Our labeled dataset is a large conjunction

$$\bigwedge_{i=1}^{m} \text{Label}(x_i, y_i)$$

The goal of learning is to make this conjunction true. Or to maximize its relaxation

Or to minimize its negative log relaxation. Using the product t-norm gives us a loss

This is the cross entropy loss

$$-\log\left[\bigwedge_{i=1}^{m} \text{Label}(x_i, y_i)\right] = -\log \prod_{i=1}^{m} \text{model}(x_i, y_i) = \boxed{-\sum_{i} \log \text{model}(x_i, y_i)}$$

Example 1: A conjunction with a different t-norm

Consider the conjunction $\alpha = X_1 \wedge X_2$ over two variables
Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α (written as $[\alpha]$).

First, we need to pick a t-norm.

Example 1: A conjunction with a different t-norm

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

Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α (written as $[\alpha]$).

First, we need to pick a t-norm. Let's choose Łukasiewicz t-norm with $a \wedge b = \max(0, a + b - 1)$

$$[\alpha] = [X_1 \wedge X_2] = \max(0, p_1 + p_2 - 1)$$

Example 1: A conjunction with a different t-norm

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

Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α (written as $[\alpha]$).

First, we need to pick a t-norm. Let's choose Łukasiewicz t-norm with $a \wedge b = \max(0, a + b - 1)$

$$[\alpha] = [X_1 \wedge X_2] = \max(0, p_1 + p_2 - 1)$$

The loss the negative of this expression (In this case, taking log need not help. Why?)

This lecture

- Quick refresher of propositional logic
- Multi-valued logics
- T-norms and their associated functions
- Logic to losses via t-norms
- Examples
  - Example 1: Conjunction
  - Example 2: Implication
- Syntax versus semantics
  - Example 3: The exactly one constraint

Example 2: Implication

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

Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α ($\text{written as } [\alpha]$).

Example 2: Implication

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

Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α (written as $[\alpha]$).

First, we need to pick a t-norm. Let's choose Product t-norm with $a \rightarrow_R b = \min\left(1, \frac{b}{a}\right)$

Example 2: Implication

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

Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α ($\text{written as } [\alpha]$).

First, we need to pick a t-norm. Let's choose Product t-norm with $a \rightarrow_R b = \min\left(1, \frac{b}{a}\right)$

We could have used a different definition of the implication. We will compare them later

Example 2: Implication

Consider the implication $\alpha = X_1 \to X_2$ over two variables
Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α (written as $[\alpha]$).

First, we need to pick a t-norm. Let's choose Product t-norm with $a \to_R b = \min\left(1, \frac{b}{a}\right)$

$$[\alpha] = [X_1 \to X_2] = \min\left(1, \frac{p_2}{p_1}\right)$$

Example 2: Implication

Consider the implication $\alpha = X_1 \rightarrow X_2$ over two variables
Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α (written as $[\alpha]$).

First, we need to pick a t-norm. Let's choose Product t-norm with $a \rightarrow_R b = \min\left(1, \frac{b}{a}\right)$

$$[\alpha] = [X_1 \rightarrow X_2] = \min\left(1, \frac{p_2}{p_1}\right)$$

Our goal: To find models that make the statement α true.

Example 2: Implication

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

Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α (written as $[\alpha]$).

First, we need to pick a t-norm. Let's choose Product t-norm with $a \to_R b = \min\left(1, \frac{b}{a}\right)$

$$[\alpha] = [X_1 \to X_2] = \min\left(1, \frac{p_2}{p_1}\right)$$

Our goal: To find models that make the statement α true.

Or equivalently: to find models that make the –ve log relaxation as low as possible

Example 2: Implication

Consider the implication $\alpha = X_1 \to X_2$ over two variables
Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α (written as $[\alpha]$).

First, we need to pick a t-norm. Let's choose Product t-norm with $a \to_R b = \min\left(1, \frac{b}{a}\right)$

$$[\alpha] = [X_1 \to X_2] = \min\left(1, \frac{p_2}{p_1}\right)$$

Our goal: To find models that make the statement α true.
Or equivalently: to find models that make the –ve log relaxation as low as possible

$$L(\alpha, p) = -\log\min\left(1, \frac{p_2}{p_1}\right) = \max(0, \log p_1 - \log p_2)$$

This loss only has a positive value when $p_1 > p_2$

Example 2: Implication

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

Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α (written as $[\alpha]$).

First, we need to pick a t-norm. Let's choose Product t-norm with $a \to_R b = \min\left(1, \frac{b}{a}\right)$

$$[\alpha] = [X_1 \to X_2] = \min\left(1, \frac{p_2}{p_1}\right)$$

Our goal: To find models that make the statement α true.

Or equivalently: to find models that make the –ve log relaxation as low as possible

$$L(\alpha, p) = -\log\min\left(1, \frac{p_2}{p_1}\right) = \max(0, \log p_1 - \log p_2)$$

This loss only has a positive value when $p_1 > p_2$

Imposes a preference that X_2 should be more true than X_1

Example 2: Implication

Consider the implication $\alpha = X_1 \rightarrow X_2$ over two variables
Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α (written as $[\alpha]$).

First, we need to pick a t-norm.

Example 2: Implication

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

Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α (written as $[\alpha]$).

First, we need to pick a t-norm. Let's choose Łukasiewicz t-norm with $a \to_R b = \min(1, 1 - a + b)$

$$[\alpha] = [X_1 \to X_2] = \min(1, 1 - p_1 + p_2)$$

Example 2: Implication

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

Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α (written as $[\alpha]$).

First, we need to pick a t-norm. Let's choose Łukasiewicz t-norm with $a \to_R b = \min(1, 1 - a + b)$

$$[\alpha] = [X_1 \to X_2] = \min(1, 1 - p_1 + p_2)$$

Our goal: To find models that make the statement α true.

Example 2: Implication

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

Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α ($\text{written as } [\alpha]$).

First, we need to pick a t-norm. Let's choose Łukasiewicz t-norm with $a \rightarrow_R b = \min(1, 1 - a + b)$

$$[\alpha] = [X_1 \rightarrow X_2] = \min(1, 1 - p_1 + p_2)$$

Our goal: To find models that make the statement α true.

Or equivalently: to find models that make the –ve relaxation as low as possible

$$L(\alpha, p) = -\min(1, 1 - p_1 + p_2) = \max(-1, p_1 - p_2 - 1)$$

Example 2: Implication

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

Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α (written as $[\alpha]$).

First, we need to pick a t-norm. Let's choose Łukasiewicz t-norm with $a \to_R b = \min(1{,}1 - a + b)$

$$[\alpha] = [X_1 \to X_2] = \min(1{,}1 - p_1 + p_2)$$

Our goal: To find models that make the statement α true.

Or equivalently: to find models that make the –ve relaxation as low as possible

$$L(\alpha, p) = -\min(1{,}1 - p_1 + p_2) = \max(-1, p_1 - p_2 - 1)$$

Since constants don't matter, we can write the loss as $L(\alpha, p) = \max(0, p_1 - p_2)$

Example 2: Implication

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

Suppose a neural network produces degrees of truth p_1 and p_2 associated with X_1 and X_2

Let us work out the t-norm losses associated with this relaxation of α (written as $[\alpha]$).

First, we need to pick a t-norm. Let's choose Łukasiewicz t-norm with $a \rightarrow_R b = \min(1, 1 - a + b)$

$$[\alpha] = [X_1 \rightarrow X_2] = \min(1, 1 - p_1 + p_2)$$

Our goal: To find models that make the statement α true.

Or equivalently: to find models that make the –ve relaxation as low as possible

$$L(\alpha, p) = -\min(1, 1 - p_1 + p_2) = \max(-1, p_1 - p_2 - 1)$$

Since constants don't matter, we can write the loss as $L(\alpha, p) = \max(0, p_1 - p_2)$

Compare to the product t-norm relaxation

$$\max(0, \log p_1 - \log p_2)$$

This lecture

- Quick refresher of propositional logic
- Multi-valued logics
- T-norms and their associated functions
- Logic to losses via t-norms
- Examples
  - Example 1: Conjunction
  - Example 2: Implication
- Syntax versus semantics
  - Example 3: The exactly one constraint

T-norms of logically equivalent expressions

Semantic loss had the same loss value irrespective of how you wrote your Boolean function

What about t-norm relaxations?

T-norms of logically equivalent expressions

Semantic loss had the same loss value irrespective of how you wrote your Boolean function

What about t-norm relaxations? Not necessarily

T-norms of logically equivalent expressions

Semantic loss had the same loss value irrespective of how you wrote your Boolean function

What about t-norm relaxations? Not necessarily

For example, consider the implication $X_1 \rightarrow X_2$, which is logically equivalent to $\neg X_1 \vee X_2$. Do these two have the same losses with the product t-norm?

T-norms of logically equivalent expressions

Semantic loss had the same loss value irrespective of how you wrote your Boolean function

What about t-norm relaxations? Not necessarily

For example, consider the implication $X_1 \rightarrow X_2$, which is logically equivalent to $\neg X_1 \vee X_2$. Do these two have the same losses with the product t-norm?

$X_1 \rightarrow X_2$:

Loss = $\max(0, \log p_1 - \log p_2)$

T-norms of logically equivalent expressions

Semantic loss had the same loss value irrespective of how you wrote your Boolean function

What about t-norm relaxations? Not necessarily

For example, consider the implication $X_1 \rightarrow X_2$, which is logically equivalent to $\neg X_1 \vee X_2$. Do these two have the same losses with the product t-norm?

$X_1 \rightarrow X_2$:

Loss = $\max(0, \log p_1 - \log p_2)$

$\neg X_1 \vee X_2$:

T-norms of logically equivalent expressions

Semantic loss had the same loss value irrespective of how you wrote your Boolean function

What about t-norm relaxations? Not necessarily

For example, consider the implication $X_1 \to X_2$, which is logically equivalent to $\neg X_1 \vee X_2$. Do these two have the same losses with the product t-norm?

$X_1 \to X_2$:

Loss = $\max(0, \log p_1 - \log p_2)$

$\neg X_1 \vee X_2$:

Loss = $-\log(1 - p_1 + p_1 p_2)$

T-norms of logically equivalent expressions

Semantic loss had the same loss value irrespective of how you wrote your Boolean function

What about t-norm relaxations? Not necessarily

For example, consider the implication $X_1 \rightarrow X_2$, which is logically equivalent to $\neg X_1 \vee X_2$. Do these two have the same losses with the product t-norm?

$X_1 \rightarrow X_2$:

Loss = $\max(0, \log p_1 - \log p_2)$

$\neg X_1 \vee X_2$:

Loss = $-\log(1 - p_1 + p_1 p_2)$

Clearly these are different functions: T-norm losses are syntactic relaxations

T-norms of logically equivalent expressions

Semantic loss had the same loss value irrespective of how you wrote your Boolean function

What about t-norm relaxations? Not necessarily

For example, consider the implication $X_1 \to X_2$, which is logically equivalent to $\neg X_1 \vee X_2$. Do these two have the same losses with the product t-norm?

$X_1 \to X_2$:

Loss = $\max(0, \log p_1 - \log p_2)$

$\neg X_1 \vee X_2$:

Loss = $-\log(1 - p_1 + p_1 p_2)$

Clearly these are different functions: T-norm losses are syntactic relaxations

The same Boolean function may have different (but often similarly behaving) loss functions depending on how the Boolean function is written

This lecture

- Quick refresher of propositional logic
- Multi-valued logics
- T-norms and their associated functions
- Logic to losses via t-norms
- Examples
  - Example 1: Conjunction
  - Example 2: Implication
- Syntax versus semantics
  - Example 3: The exactly one constraint

A simple semi-supervised learning example

Suppose we have:

- A small number of labeled examples for a task with k labels
- A large collection of unlabeled examples

What information can the unlabeled examples provide to a model?

An unlabeled example must also have one and exactly one of the k labels

Can this information help train a model?

The exactly-one constraint

We have seen this before

Suppose we have three possible decisions produced by one or more neural networks: X_1, X_2, X_3

We want to enforce the following constraints about these decisions:

- One of these three decisions must be `true`

$$X_1 \lor X_2 \lor X_3$$

- No two of the three decisions can simultaneously be `true`

$$\neg X_1 \lor \neg X_2$$
$$\neg X_2 \lor \neg X_3$$
$$\neg X_3 \lor \neg X_1$$

Together, these constraints require that exactly one of the decisions should be `true`

How can we incorporate this knowledge into our loss?

The compiled exactly-one constraint

The constraint

$$(X_1 \vee X_2 \vee X_3) \wedge (\neg X_1 \vee \neg X_2) \wedge (\neg X_2 \vee \neg X_3) \wedge (\neg X_3 \vee \neg X_1)$$

Let’s derive the loss

The exactly-one constraint written differently

The constraint can be equivalently written as

$$(X_1 \wedge \neg X_2 \wedge \neg X_3) \vee (\neg X_1 \wedge X_2 \wedge \neg X_3) \vee (\neg X_1 \wedge \neg X_2 \wedge X_3)$$

Let’s derive the loss

Syntactic choices may matter

The two versions of the losses we derived are not identical

Which one do we pick?

Here, understanding the optimizer is helpful. Generally, it seems be better to pick a version that produces a loss with that is a big sum of logs rather than the log of sums

Summary: T-norm losses

T-norms

- Systematic extension of propositional logic to use infinite degrees of truth
- A t-norm is a generalization of a conjunction
- T-conorms and residuum generalize the disjunction and the implication

Key technical component

- Treat model predictions as degrees of truth
- Compile a Boolean formula using a chosen t-norm
- Minimize the relaxation (or its logarithm) to find a model that is consistent with the logic

Pros and cons

- No computational issues such as model counting
- But syntactic variations in how we write the constraints may produce different loss expressions