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

Logic as loss: Semantic loss

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

A second look at the semantic loss

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

Is there a catch?

A second look at the semantic loss

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

This term requires us to accumulate quantities computed for every satisfying assignment for the formula α

A second look at the semantic loss

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

This term requires us to accumulate quantities computed for every satisfying assignment for the formula α

A second look at the semantic loss

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

This term requires us to accumulate quantities computed for every satisfying assignment for the formula α

Is this a problem? Have we seen this before?

Weighted model counting

Recall: model = a satisfying assignment to a propositional formula

Weighted model counting

Recall: model = a satisfying assignment to a propositional formula

Model counting: How many satisfying assignments does the propositional formula have?

Weighted model counting

Recall: model = a satisfying assignment to a propositional formula

Model counting: How many satisfying assignments does the propositional formula have?

Example: $X_1 \rightarrow X_2$

Weighted model counting

Recall: model = a satisfying assignment to a propositional formula

Model counting: How many satisfying assignments does the propositional formula have?

Example: $X_1 \rightarrow X_2$

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

Weighted model counting

Recall: model = a satisfying assignment to a propositional formula

Model counting: How many satisfying assignments does the propositional formula have?

Example: $X_1 \rightarrow X_2$

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

Not a model for this formula

Weighted model counting

Recall: model = a satisfying assignment to a propositional formula

Model counting: How many satisfying assignments does the propositional formula have?

Example: $X_1 \to X_2$

X_1	X_2	α
$\top$	$\top$	$\top$
$\cancel{\top}$	$\cancel{\bot}$	$\cancel{\bot}$
$\bot$	$\top$	$\top$
$\bot$	$\bot$	$\top$

Number of models = 3

Weighted model counting

Recall: model = a satisfying assignment to a propositional formula

Model counting: How many satisfying assignments does the propositional formula have?

This is a #P problem

Example: $X_1 \rightarrow X_2$

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

Number of models = 3

Weighted model counting

Recall: model = a satisfying assignment to a propositional formula

Model counting: How many satisfying assignments does the propositional formula have?

This is a #P problem

Weighted model counting

Each assignment to a variable has weights

Example: $X_1 \rightarrow X_2$

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

Number of models = 3

Weighted model counting

Recall: model = a satisfying assignment to a propositional formula

Model counting: How many satisfying assignments does the propositional formula have?

This is a #P problem

Weighted model counting

Each assignment to a variable has weights

Variable weights

X	$w(X)$	$w(\neg X)$
X_1	3	2
X_2	5	7

Example: $X_1 \rightarrow X_2$

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

Number of models = 3

Weighted model counting

Recall: model = a satisfying assignment to a propositional formula

Model counting: How many satisfying assignments does the propositional formula have?

- This is a #P problem

Weighted model counting

- Each assignment to a variable has weights
- The weight of a model is the product of the variable weights

Variable weights

X	$w(X)$	$w(\neg X)$
X_1	3	2
X_2	5	7

Example: $X_1 \rightarrow X_2$

X_1	X_2	α
$\top$	$\top$	$\top$
$\cancel{\top}$	$\cancel{\bot}$	$\cancel{\bot}$
$\bot$	$\top$	$\top$
$\bot$	$\bot$	$\top$

Number of models = 3

Weighted model counting

Recall: model = a satisfying assignment to a propositional formula

Model counting: How many satisfying assignments does the propositional formula have?

- This is a #P problem

Weighted model counting

- Each assignment to a variable has weights
- The weight of a model is the product of the variable weights

Variable weights

X	$w(X)$	$w(\neg X)$
X_1	3	2
X_2	5	7

Example: $X_1 \rightarrow X_2$

X_1	X_2	α	weight
⊤	⊤	⊤	$3 \times 5 = 15$
~~⊤~~	~~⊥~~	~~⊥~~	~~$3 \times 7 = 21$~~
⊥	⊤	⊤	$2 \times 5 = 10$
⊥	⊥	⊤	$2 \times 7 = 14$

Number of models = 3

Weighted model counting

Recall: model = a satisfying assignment to a propositional formula

Model counting: How many satisfying assignments does the propositional formula have?

- This is a #P problem

Weighted model counting

- Each assignment to a variable has weights
- The weight of a model is the product of the variable weights
- Our goal is to add up model weights of all satisfying assignments

Variable weights

X	$w(X)$	$w(\neg X)$
X_1	3	2
X_2	5	7

Example: $X_1 \rightarrow X_2$

X_1	X_2	α	weight
⊤	⊤	⊤	$3\times5 = 15$
~~⊤~~	~~⊥~~	~~⊥~~	~~$3\times7 = 21$~~
⊥	⊤	⊤	$2\times5 = 10$
⊥	⊥	⊤	$2\times7 = 14$

Number of models = 3

Weighted model counting

Recall: model = a satisfying assignment to a propositional formula

Model counting: How many satisfying assignments does the propositional formula have?

- This is a #P problem

Weighted model counting

- Each assignment to a variable has weights
- The weight of a model is the product of the variable weights
- Our goal is to add up model weights of all satisfying assignments

Variable weights

X	$w(X)$	$w(\neg X)$
X_1	3	2
X_2	5	7

Example: $X_1 \rightarrow X_2$

X_1	X_2	α	weight
$\top$	$\top$	$\top$	$3\times 5 = \boxed{15}$
~~$\top$~~	~~$\bot$~~	~~$\bot$~~	~~$3\times 7 = 21$~~
$\bot$	$\top$	$\top$	$2\times 5 = \boxed{10}$
$\bot$	$\bot$	$\top$	$2\times 7 = \boxed{14}$

Number of models = 3

Weighted model count = 15 + 10 + 14 = 39

Weighted model counting

Recall: model = a satisfying assignment to a propositional formula

Model counting: How many satisfying assignments does the propositional formula have?

- This is a #P problem

Weighted model counting

- Each assignment to a variable has weights
- The weight of a model is the product of the variable weights
- Our goal is to add up model weights of all satisfying assignments
- Also #P

Variable weights

X	$w(X)$	$w(\neg X)$
X_1	3	2
X_2	5	7

Example: $X_1 \rightarrow X_2$

X_1	X_2	α	weight
$\top$	$\top$	$\top$	$3 \times 5 = 15$
~~$\top$~~	~~$\bot$~~	~~$\bot$~~	~~$3 \times 7 = 21$~~
$\bot$	$\top$	$\top$	$2 \times 5 = 10$
$\bot$	$\bot$	$\top$	$2 \times 7 = 14$

Number of models = 3

Weighted model count = 15 + 10 + 14 = 39

A second look at the semantic loss

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

This term requires us to accumulate quantities computed for every satisfying assignment for the formula α

Is this a problem? Have we seen this before?

Computing the semantic loss requires us to perform weighted model counting

Intractable in the worst case, *but tractable subsets of logic exist*

Logic as loss: Semantic loss

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

A model counting example

How many satisfying assignments does this formula have?

$$\alpha = X_1 \land \neg(X_2 \leftrightarrow X_3)$$

A model counting example

How many satisfying assignments does this formula have?

$$\alpha = X_1 \land \neg(X_2 \leftrightarrow X_3)$$

One way to count:

Enumerate all satisfying assignments

X_1	X_2	X_3	α
$\top$	$\top$	$\top$	$\bot$
$\top$	$\top$	$\bot$	$\top$
$\top$	$\bot$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\top$	$\bot$
$\bot$	$\top$	$\bot$	$\bot$
$\bot$	$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$

A model counting example

How many satisfying assignments does this formula have?

$$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$$

The formula has two satisfying assignments

One way to count:

Enumerate all satisfying assignments

X_1	X_2	X_3	α
$\top$	$\top$	$\top$	$\bot$
$\top$	$\top$	$\bot$	$\top$
$\top$	$\bot$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\top$	$\bot$
$\bot$	$\top$	$\bot$	$\bot$
$\bot$	$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$

A model counting example

How many satisfying assignments does this formula have?

$$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$$

The formula has two satisfying assignments

One way to count:

Enumerate all satisfying assignments

Can we do better?

In the worst case, not really

X_1	X_2	X_3	α
$\top$	$\top$	$\top$	$\perp$
$\top$	$\top$	$\perp$	$\top$
$\top$	$\perp$	$\top$	$\top$
$\top$	$\perp$	$\perp$	$\perp$
$\perp$	$\top$	$\top$	$\perp$
$\perp$	$\top$	$\perp$	$\perp$
$\perp$	$\perp$	$\top$	$\perp$
$\perp$	$\perp$	$\perp$	$\perp$

A model counting example

How many satisfying assignments does this formula have?

$$\alpha = X_1 \land \neg(X_2 \leftrightarrow X_3)$$

The formula has two satisfying assignments

X_1	X_2	X_3	α
$\top$	$\top$	$\top$	$\bot$
$\top$	$\top$	$\bot$	$\top$
$\top$	$\bot$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\top$	$\bot$
$\bot$	$\top$	$\bot$	$\bot$
$\bot$	$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$

One way to count:

Enumerate all satisfying assignments

Can we do better?

In the worst case, not really

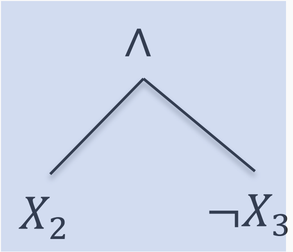
But are there easy cases?

An equivalent expression

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

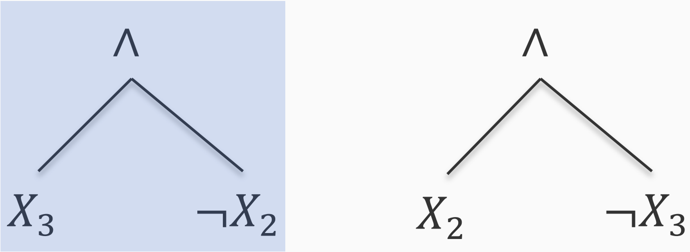
We can write this as an expression tree

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$



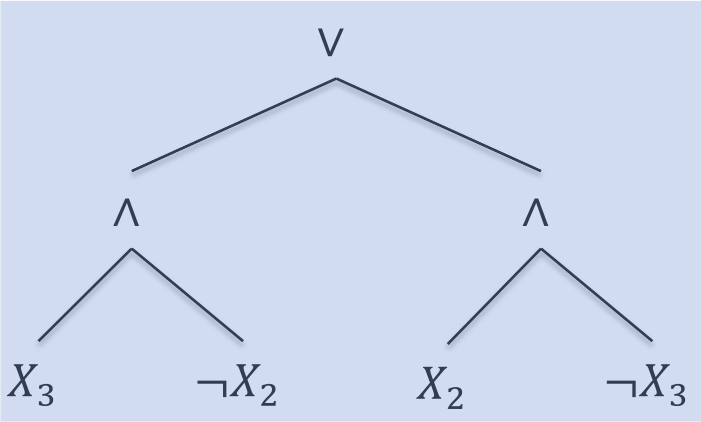
We can write this as an expression tree

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$



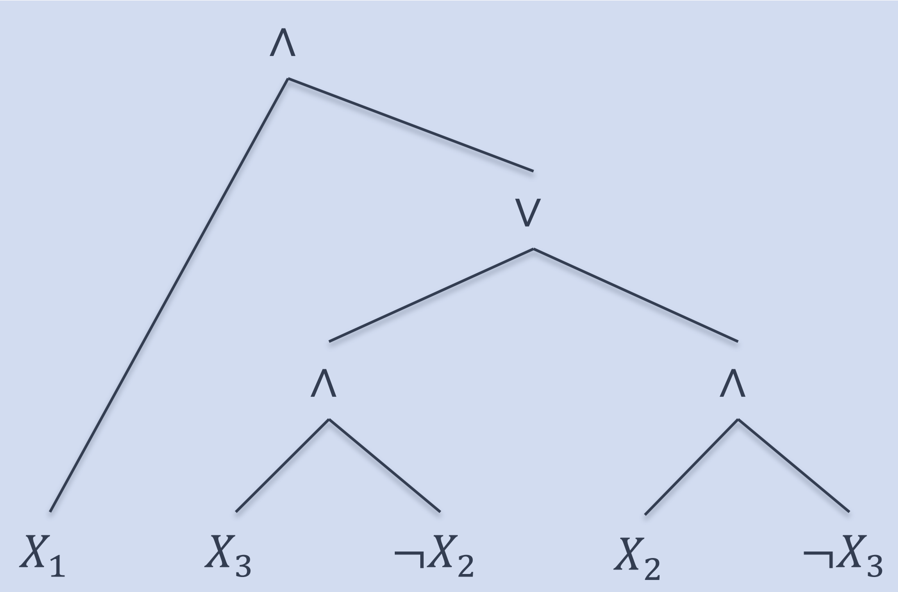
We can write this as an expression tree

$\alpha = X_1 \land \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \land \big((X_2 \land \neg X_3) \lor (\neg X_2 \land X_3)\big)$



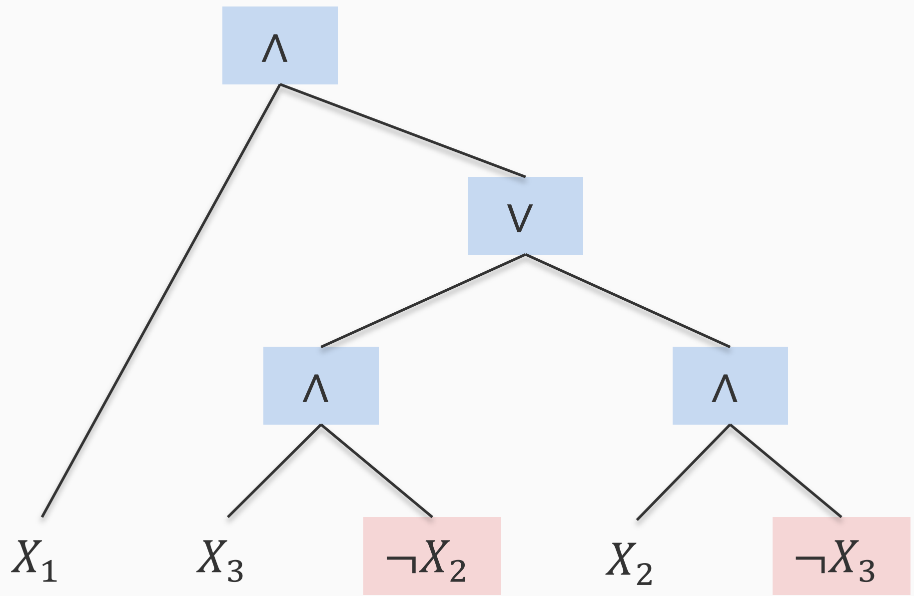
We can write this as an expression tree

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$



This expression is in the *Negation Normal Form*

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$



The only negations are in the leaves

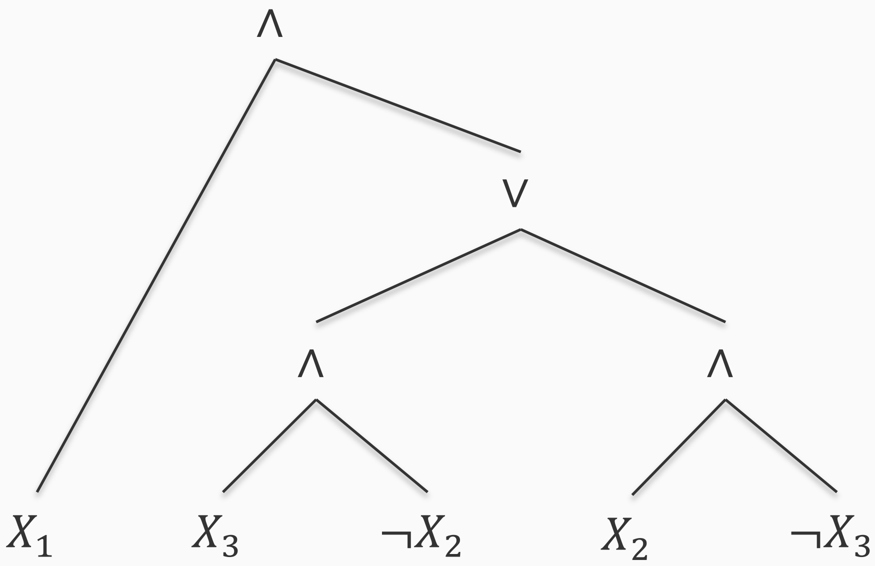
That is, the leaves are literals

All other nodes are either conjunctions or disjunctions

This expression is *Decomposable*

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

✓ Negation Normal Form

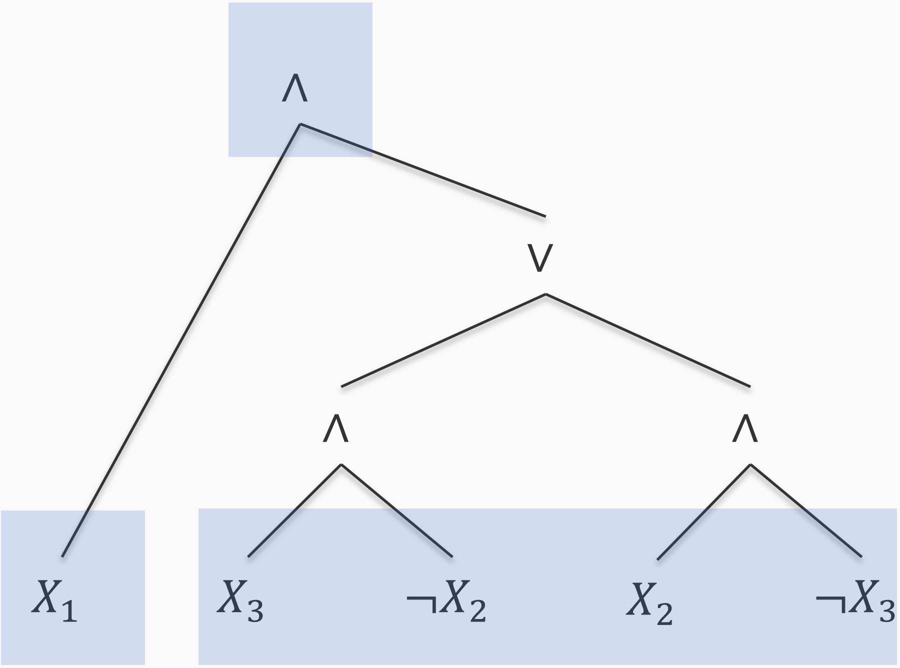


For every conjunction, the conjuncts do not share any variables

This expression is *Decomposable*

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

✓ Negation Normal Form



No overlap in variables on two sides

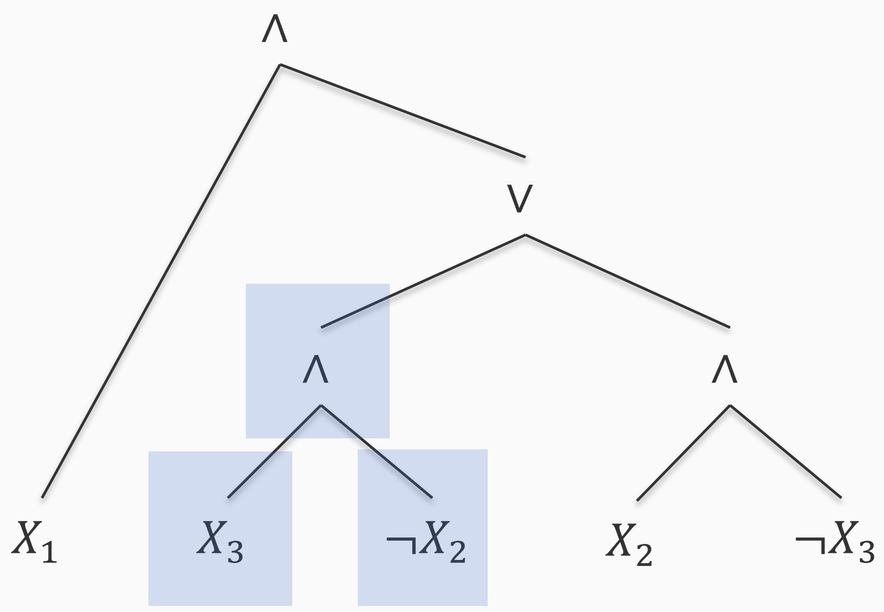
For every conjunction, the conjuncts do not share any variables

This expression is *Decomposable*

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

✓ Negation Normal Form

For every conjunction, the conjuncts do not share any variables



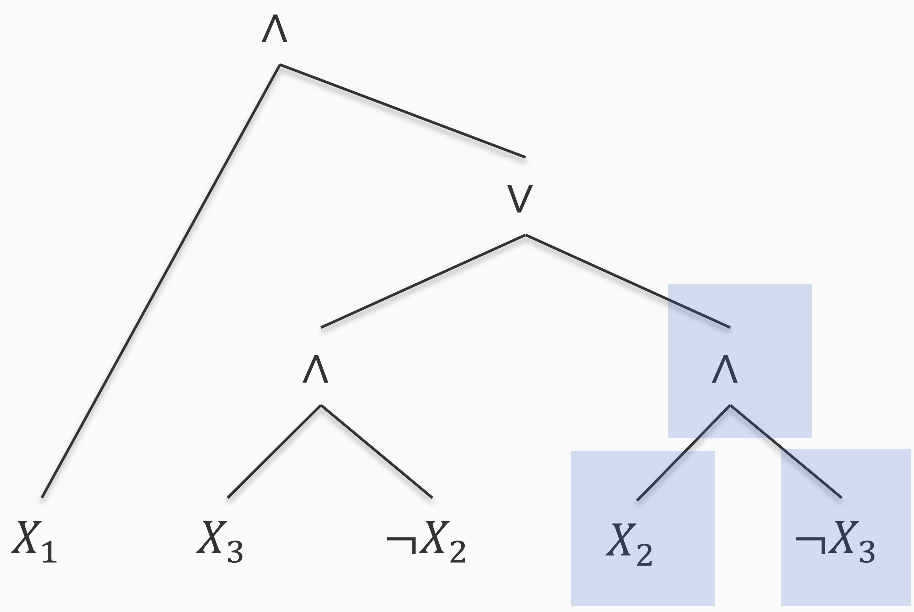
No overlap in variables on two sides

This expression is *Decomposable*

$\alpha = X_1 \land \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \land \left((X_2 \land \neg X_3) \lor (\neg X_2 \land X_3)\right)$

✓ Negation Normal Form

For every conjunction, the conjuncts do not share any variables

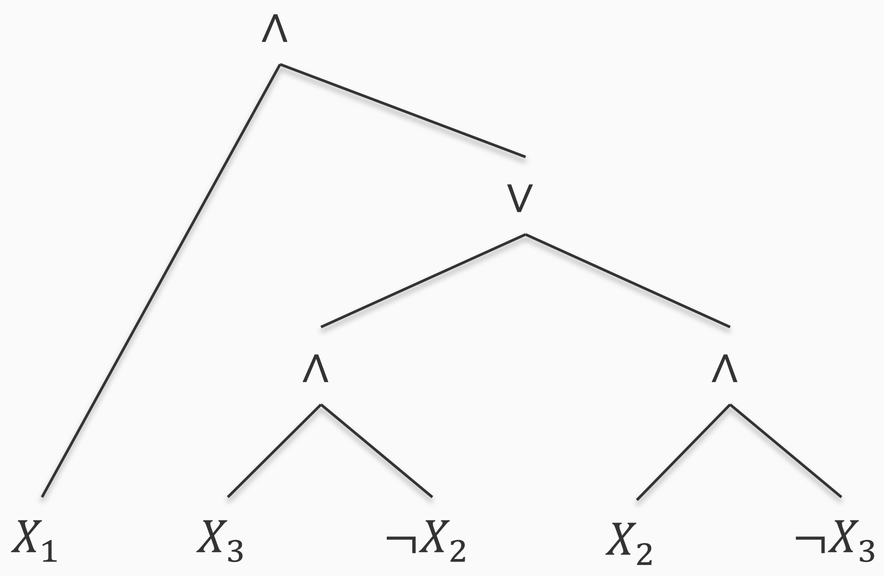


No overlap in variables on two sides

This expression is *deterministic*

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

- ✓ Negation Normal Form
- ✓ Decomposable

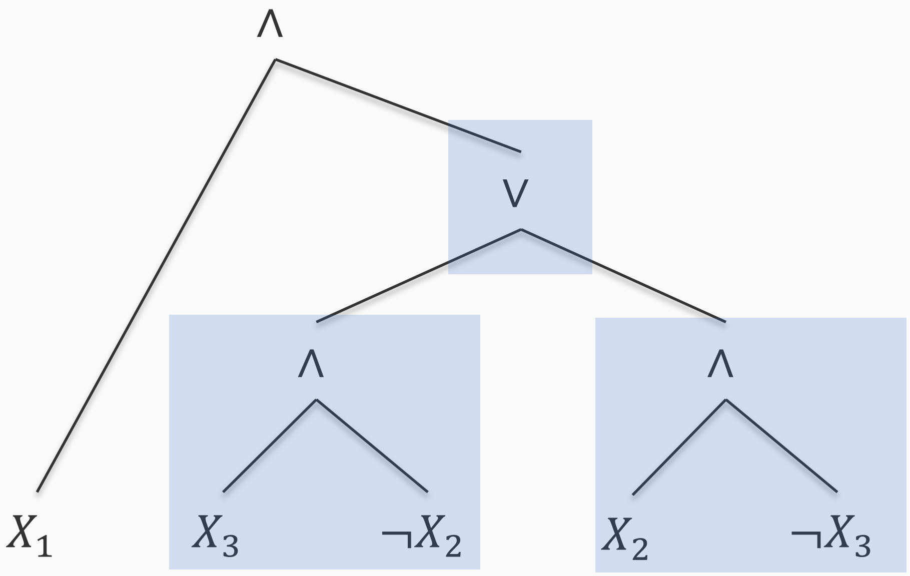


For every disjunction, the disjuncts contradict each other

This expression is *deterministic*

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

- ✓ Negation Normal Form
- ✓ Decomposable

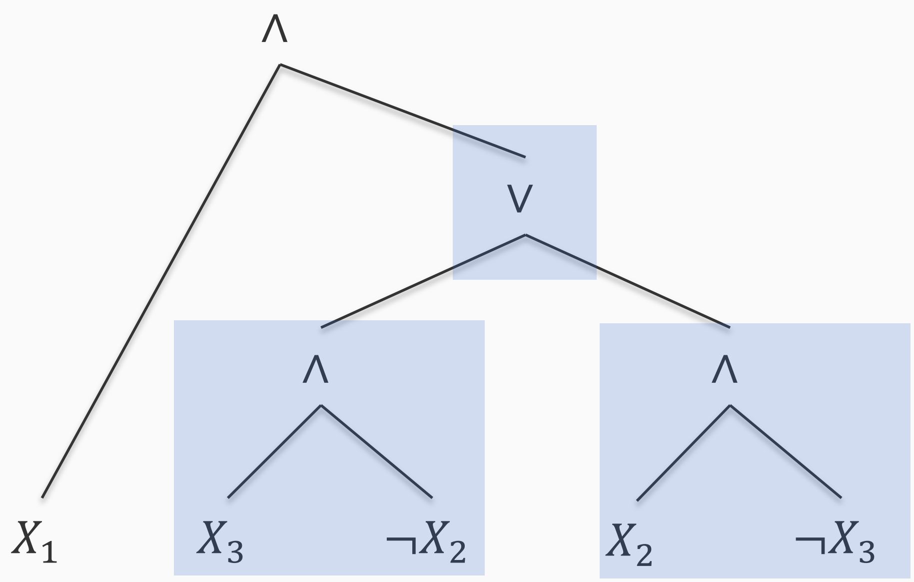


For every disjunction, the disjuncts contradict each other

This expression is *deterministic*

$\alpha = X_1 \land \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \land \big((X_2 \land \neg X_3) \lor (\neg X_2 \land X_3)\big)$

- ✓ Negation Normal Form
- ✓ Decomposable



For every disjunction, the disjuncts contradict each other

The left side of the disjunction is $X_2 \land \neg X_3$

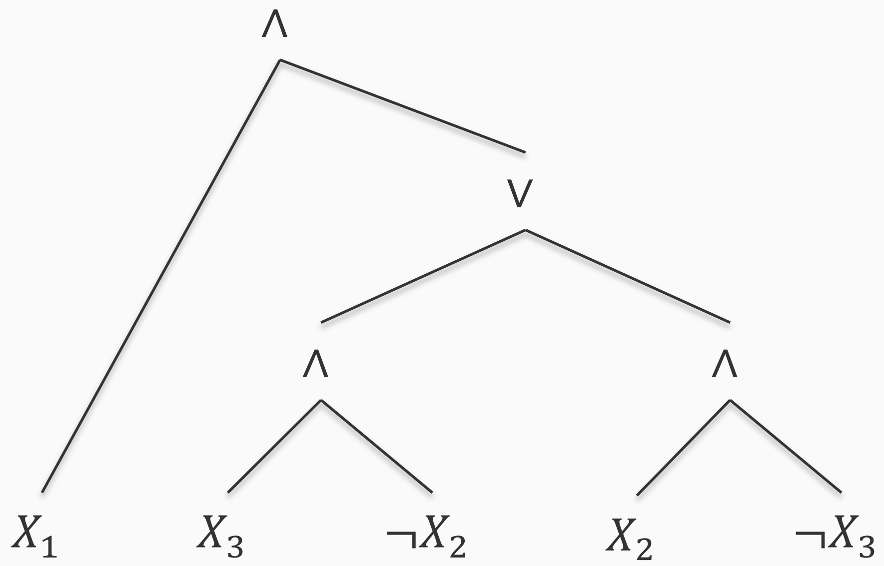
The right side is $\neg X_2 \land X_3$

Both these expressions cannot be simultaneously true
(you should verify this)

We have a new normal form

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

- ✓ Negation Normal Form
- ✓ Decomposable
- ✓ deterministic



With all these three properties, this expression is in a form called deterministic decomposable negation normal form (d-DNNF).

We have seen normal forms of propositional formulas before

Disjunctive normal forms,

Conjunctive normal forms

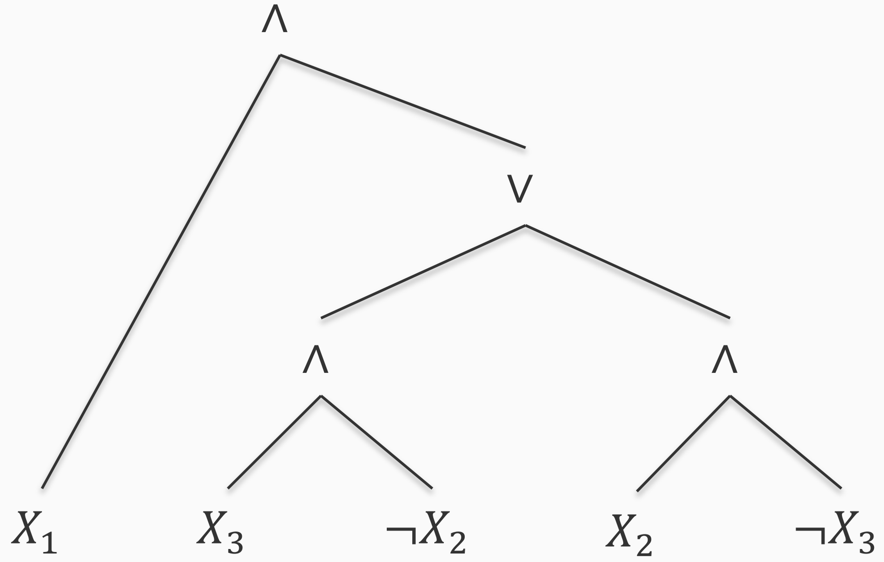
Darwiche, Adnan, and Pierre Marquis. "A knowledge compilation map." *Journal of Artificial Intelligence Research* 17 (2002)

Darwiche, Adnan. "On the tractable counting of theory models and its application to truth maintenance and belief revision." *Journal of Applied Non-Classical Logics* 11, no. 1-2 (2001): 11-34.

What makes a d-DNNF special?

$\alpha = X_1 \land \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \land \big((X_2 \land \neg X_3) \lor (\neg X_2 \land X_3)\big)$

- ✓ Negation Normal Form
- ✓ Decomposable
- ✓ deterministic



It allows us to perform model counting and weighted model counting with one traversal of the tree

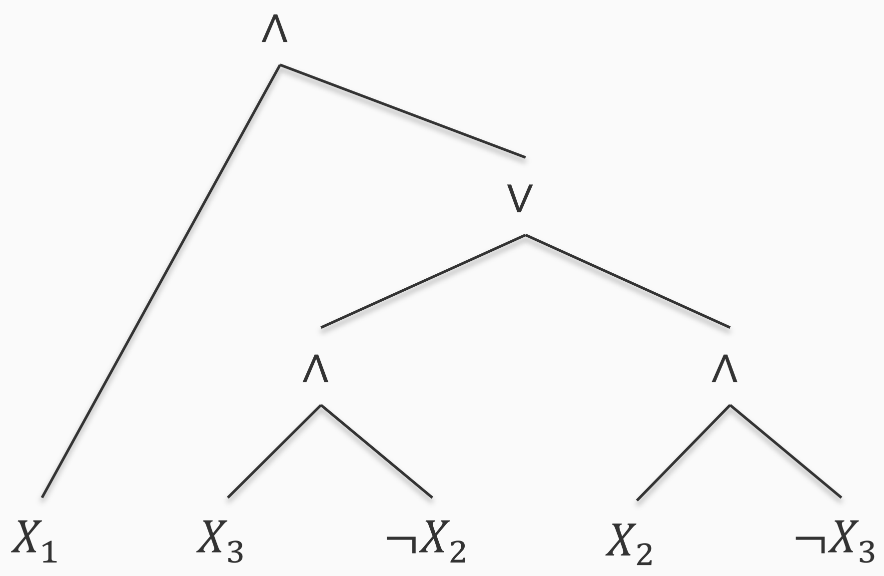
Let’s see how

What makes a d-DNNF special?

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

Recall that the original expression
had two satisfying assignments

- ✓ Negation Normal Form
- ✓ Decomposable
- ✓ deterministic



It allows us to perform model counting and weighted model counting with one traversal of the tree

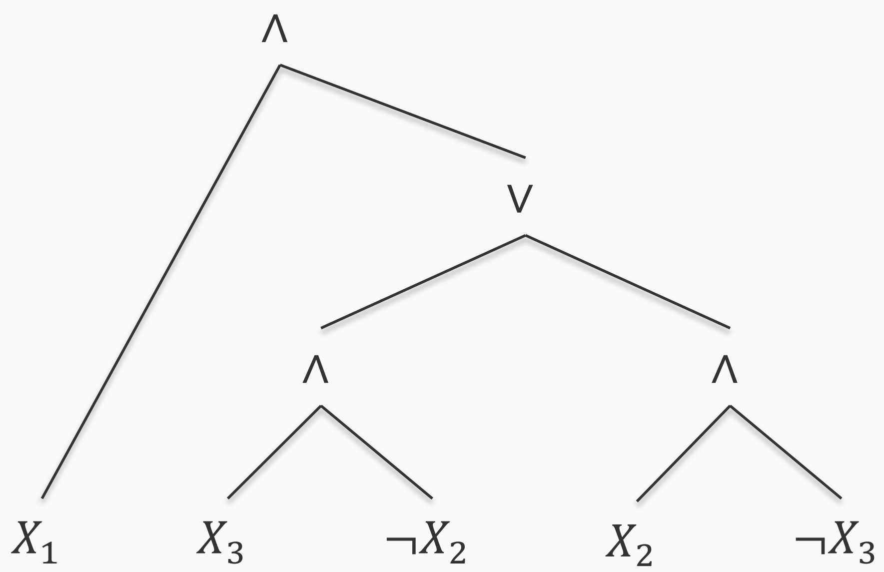
Let’s see how

What makes a d-DNNF special?

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

Recall that the original expression
had two satisfying assignments

- ✓ Negation Normal Form
- ✓ Decomposable
- ✓ deterministic



Let us construct a counting tree, where all conjunctions are replaced with a product and all disjunctions are replaced with a sum

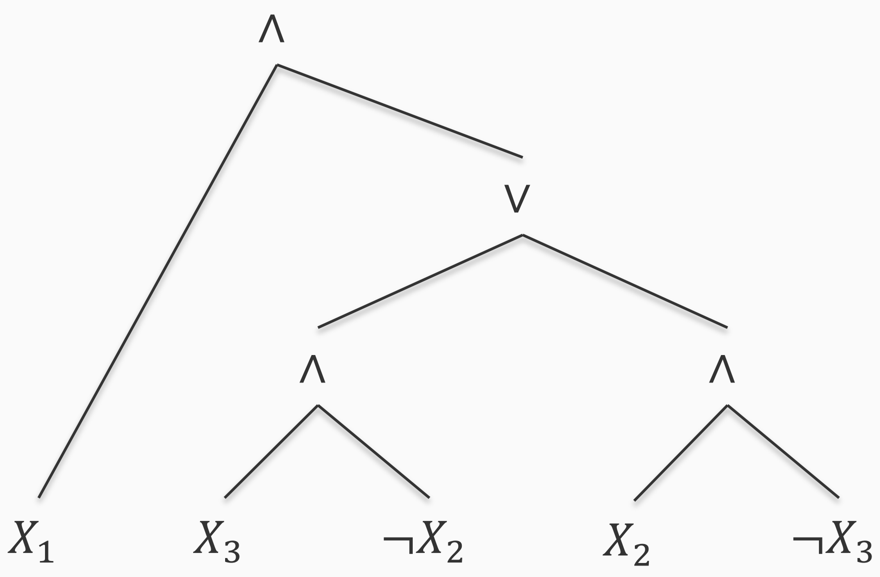
What makes a d-DNNF special?

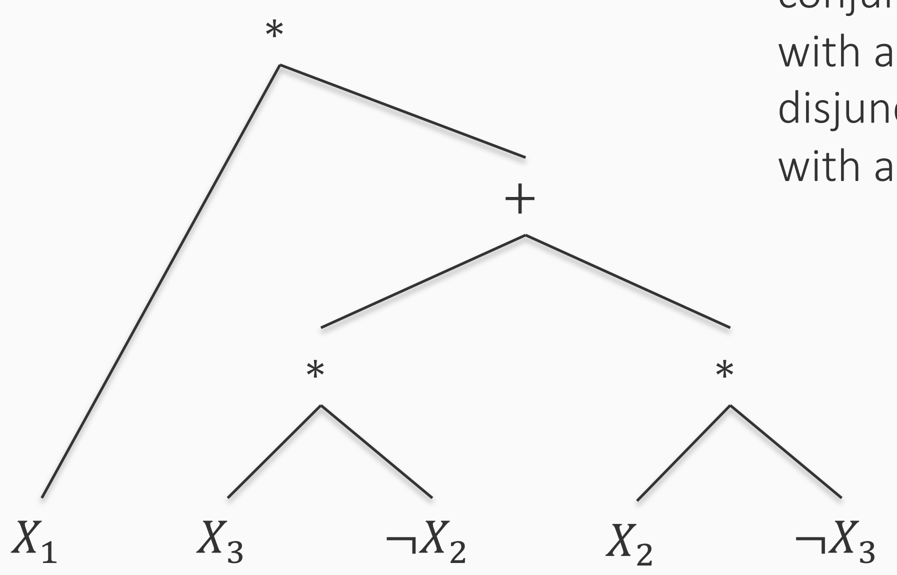
$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

Recall that the original expression
had two satisfying assignments

- ✓ Negation Normal Form
- ✓ Decomposable
- ✓ deterministic

Let us construct a counting tree, where all conjunctions are replaced with a product and all disjunctions are replaced with a sum





What makes a d-DNNF special?

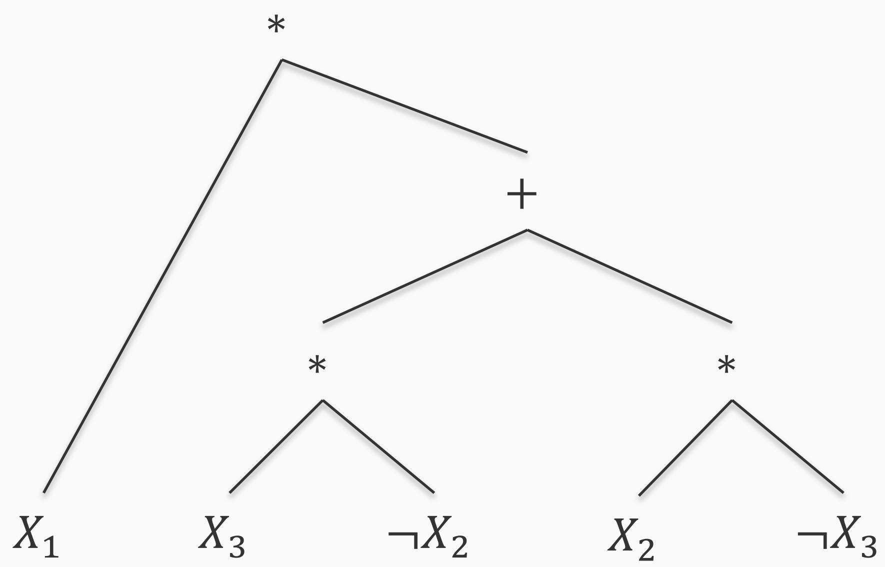
$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

Recall that the original expression
had two satisfying assignments

- ✓ Negation Normal Form
- ✓ Decomposable
- ✓ deterministic

Let us construct a counting tree, where all conjunctions are replaced with a product and all disjunctions are replaced with a sum

In the counting tree, assign every leaf node to take the value one…



What makes a d-DNNF special?

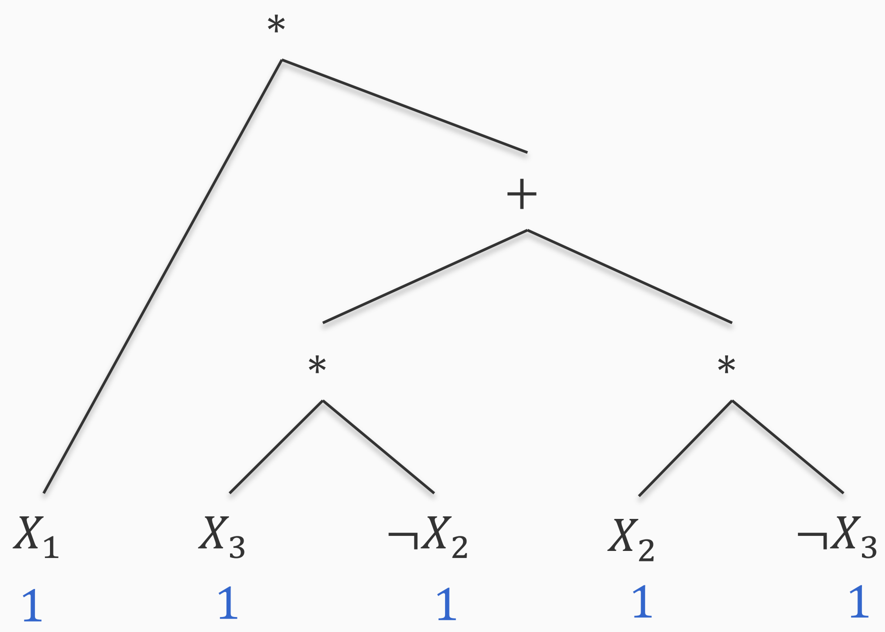
$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

Recall that the original expression
had two satisfying assignments

- ✓ Negation Normal Form
- ✓ Decomposable
- ✓ deterministic

Let us construct a counting tree, where all conjunctions are replaced with a product and all disjunctions are replaced with a sum

In the counting tree, assign every leaf node to take the value one…

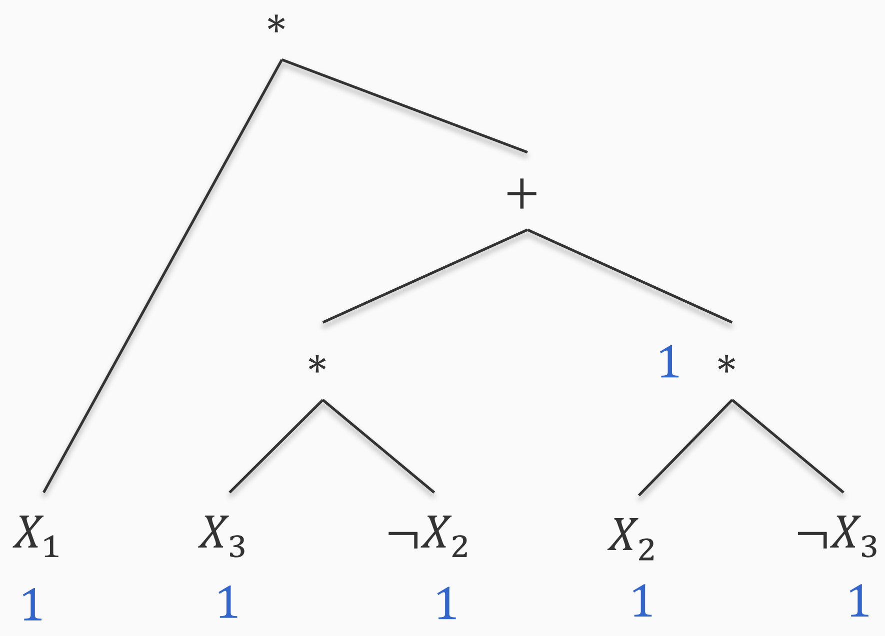


What makes a d-DNNF special?

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

Recall that the original expression
had two satisfying assignments

- ✓ Negation Normal Form
- ✓ Decomposable
- ✓ deterministic



Let us construct a counting tree, where all conjunctions are replaced with a product and all disjunctions are replaced with a sum

In the counting tree, assign every leaf node to take the value one…

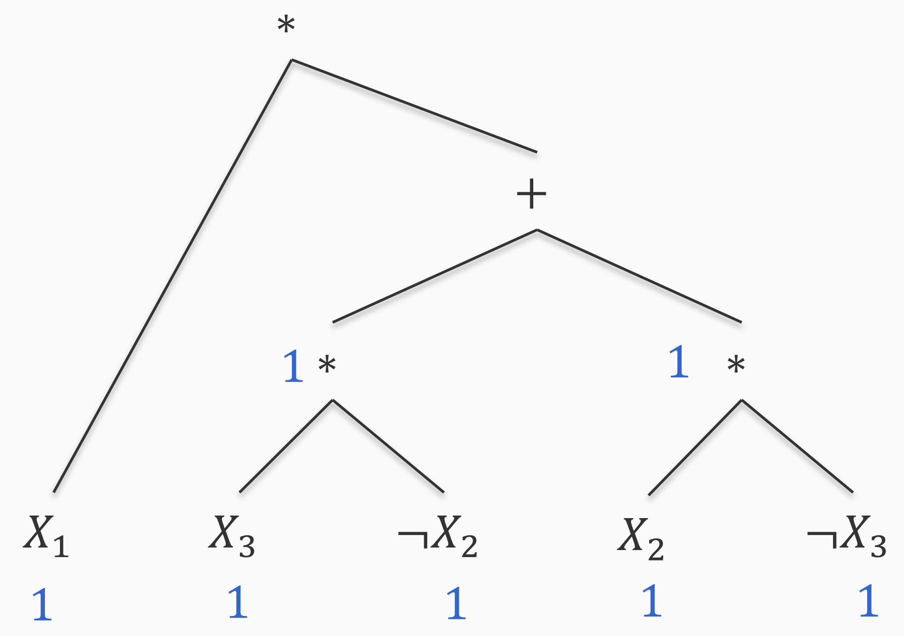
…and calculate the value of the root

What makes a d-DNNF special?

$\alpha = X_1 \land \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \land \big((X_2 \land \neg X_3) \lor (\neg X_2 \land X_3)\big)$

Recall that the original expression
had two satisfying assignments

- ✓ Negation Normal Form
- ✓ Decomposable
- ✓ deterministic



Let us construct a counting tree, where all conjunctions are replaced with a product and all disjunctions are replaced with a sum

In the counting tree, assign every leaf node to take the value one…

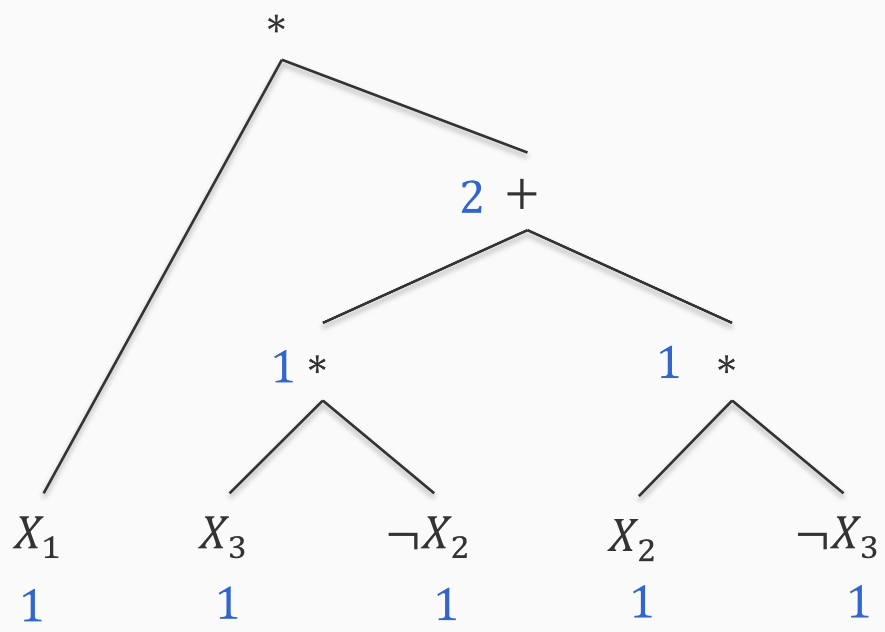
…and calculate the value of the root

What makes a d-DNNF special?

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

Recall that the original expression
had two satisfying assignments

- ✓ Negation Normal Form
- ✓ Decomposable
- ✓ deterministic



Let us construct a counting tree, where all conjunctions are replaced with a product and all disjunctions are replaced with a sum

In the counting tree, assign every leaf node to take the value one…

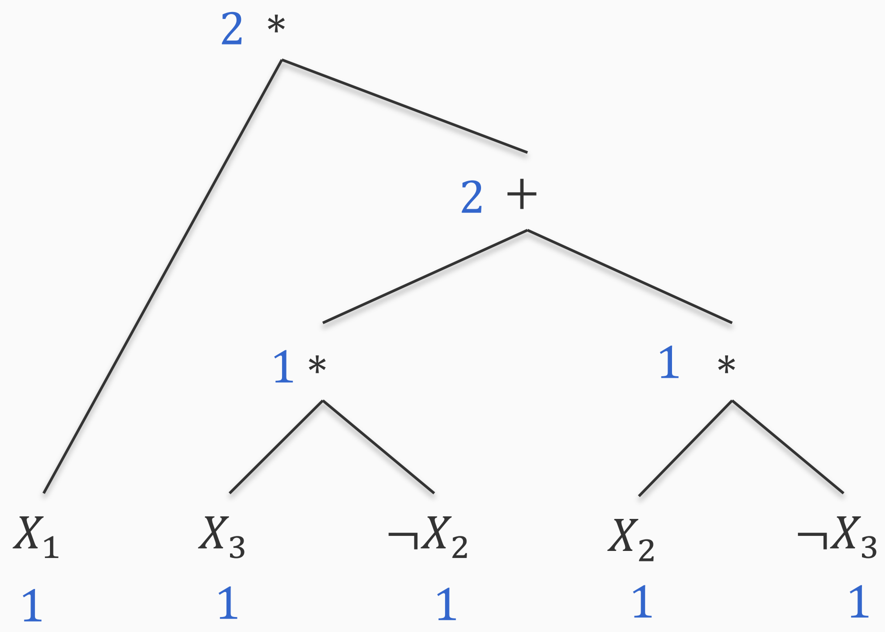
…and calculate the value of the root

What makes a d-DNNF special?

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

Recall that the original expression
had two satisfying assignments

- ✓ Negation Normal Form
- ✓ Decomposable
- ✓ deterministic



Let us construct a counting tree, where all conjunctions are replaced with a product and all disjunctions are replaced with a sum

In the counting tree, assign every leaf node to take the value one…

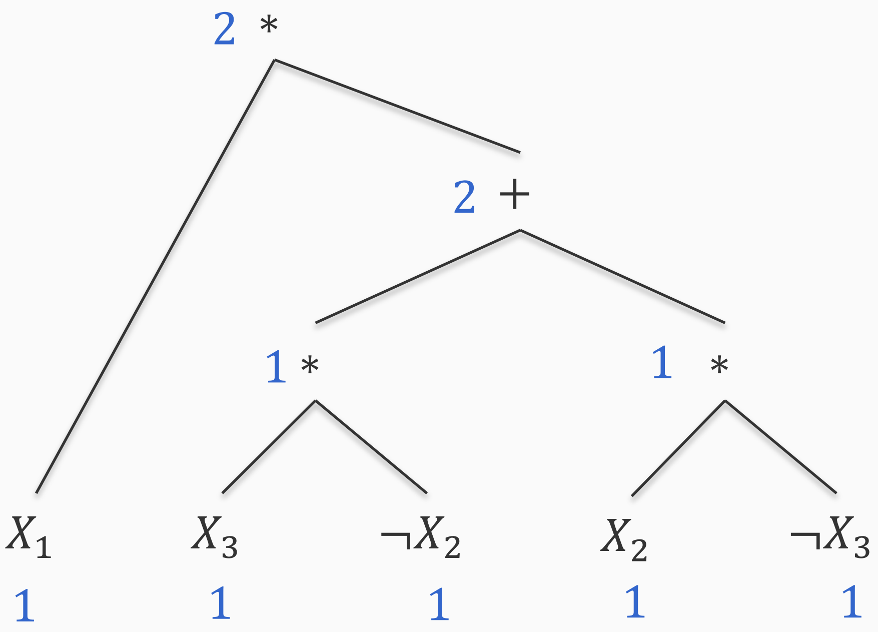
…and calculate the value of the root

What makes a d-DNNF special?

$\alpha = X_1 \land \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \land \big((X_2 \land \neg X_3) \lor (\neg X_2 \land X_3)\big)$

Recall that the original expression had two satisfying assignments

- ✓ Negation Normal Form
- ✓ Decomposable
- ✓ deterministic



Let us construct a counting tree, where all conjunctions are replaced with a product and all disjunctions are replaced with a sum

In the counting tree, assign every leaf node to take the value one…

…and calculate the value of the root

The value at the root is the number of satisfying assignments of the formula!

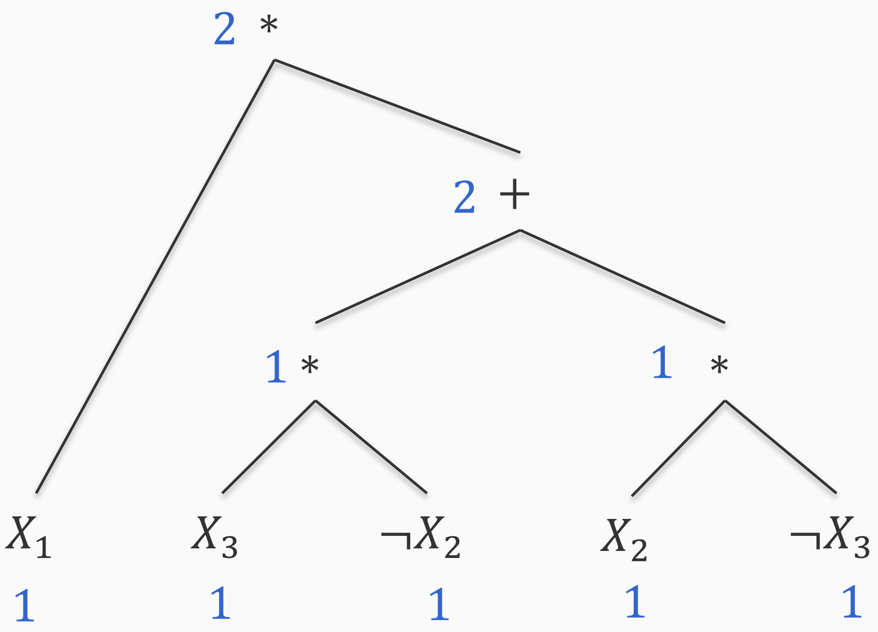
Darwiche, Adnan. "On the tractable counting of theory models and its application to truth maintenance and belief revision." *Journal of Applied Non-Classical Logics* 11, no. 1-2 (2001): 11-34.

What makes a d-DNNF special?

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

Recall that the original expression had two satisfying assignments

- ✓ Negation Normal Form
- ✓ Decomposable
- ✓ deterministic



Let us construct a counting tree, where all conjunctions are replaced with a product and all disjunctions are replaced with a sum

In the counting tree, assign every leaf node to take the value one…

…and calculate the value of the root

The value at the root is the number of satisfying assignments of the formula!

There are more nuances. See the paper for details

Darwiche, Adnan. "On the tractable counting of theory models and its application to truth maintenance and belief revision." *Journal of Applied Non-Classical Logics* 11, no. 1-2 (2001): 11-34.

What makes a d-DNNF special?

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

Recall that the original expression
had two satisfying assignments

✓ Negation Normal Form
✓ Decomposable
✓ deterministic

Let us construct a counting tree, where all conjunctions are replaced with a product and all disjunctions are replaced with a sum

Key point: We can count the number of satisfying assignments with what looks like a forward pass of a neural network

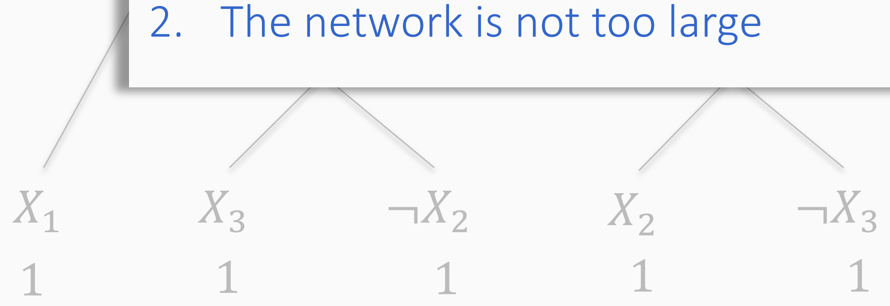
Efficient, if:
1. The network (i.e. the d-DNNF) can be constructed efficiently
2. The network is not too large

gn every leaf node to take

of the root

e number of satisfying assignments of the formula!

There are more nuances. See the paper for details



Darwiche, Adnan. "On the tractable counting of theory models and its application to truth maintenance and belief revision." *Journal of Applied Non-Classical Logics* 11, no. 1-2 (2001): 11-34.

We can also perform weighted model counting

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

Suppose we have weights for variables

$\boldsymbol{X}$	$\mathbf{w}\text{eight}(\boldsymbol{X})$	$\mathbf{w}\text{eight}(\neg\boldsymbol{X})$
X_1	0.8	0.2
X_2	0.3	0.7
X_3	0.6	0.4

Darwiche, Adnan. "On the tractable counting of theory models and its application to truth maintenance and belief revision." *Journal of Applied Non-Classical Logics* 11, no. 1-2 (2001): 11-34.

We can also perform weighted model counting

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

Suppose we have weights for variables

$\boldsymbol{X}$	$\mathbf{weight}(\boldsymbol{X})$	$\mathbf{weight}(\neg\boldsymbol{X})$
X_1	0.8	0.2
X_2	0.3	0.7
X_3	0.6	0.4

Let’s look at the truth table first

X_1	X_2	X_3	α
$\top$	$\top$	$\top$	$\bot$
$\top$	$\top$	$\bot$	$\top$
$\top$	$\bot$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\top$	$\bot$
$\bot$	$\top$	$\bot$	$\bot$
$\bot$	$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$

Darwiche, Adnan. "On the tractable counting of theory models and its application to truth maintenance and belief revision." *Journal of Applied Non-Classical Logics* 11, no. 1-2 (2001): 11-34.

We can also perform weighted model counting

$$\alpha = X_1 \land \neg(X_2 \leftrightarrow X_3) \text{ is the same as } X_1 \land \big((X_2 \land \neg X_3) \lor (\neg X_2 \land X_3)\big)$$

Suppose we have weights for variables

$\boldsymbol{X}$	weight($\boldsymbol{X}$)	weight($\neg\boldsymbol{X}$)
X_1	0.8	0.2
X_2	0.3	0.7
X_3	0.6	0.4

Let's look at the truth table first

X_1	X_2	X_3	α
$\top$	$\top$	$\top$	$\bot$
$\top$	$\top$	$\bot$	$\top$
$\top$	$\bot$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\top$	$\bot$
$\bot$	$\top$	$\bot$	$\bot$
$\bot$	$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$

Only these two rows matter because they are the only assignments that make the formula `true`

Darwiche, Adnan. "On the tractable counting of theory models and its application to truth maintenance and belief revision." *Journal of Applied Non-Classical Logics* 11, no. 1-2 (2001): 11-34.

We can also perform weighted model counting

$$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3) \text{ is the same as } X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$$

Suppose we have weights for variables

$\boldsymbol{X}$	$\mathbf{weight}(\boldsymbol{X})$	$\mathbf{weight}(\neg\boldsymbol{X})$
X_1	0.8	0.2
X_2	0.3	0.7
X_3	0.6	0.4

Let's look at the truth table first

X_1	X_2	X_3	α	weight
$\top$	$\top$	$\top$	$\bot$	
$\top$	$\top$	$\bot$	$\top$	$0.8 \times 0.3 \times 0.4 = 0.096$
$\top$	$\bot$	$\top$	$\top$	$0.8 \times 0.7 \times 0.6 = 0.336$
$\top$	$\bot$	$\bot$	$\bot$	
$\bot$	$\top$	$\top$	$\bot$	
$\bot$	$\top$	$\bot$	$\bot$	
$\bot$	$\bot$	$\top$	$\bot$	
$\bot$	$\bot$	$\bot$	$\bot$	

Only these two rows matter because they are the only assignments that make the formula `true`

Darwiche, Adnan. "On the tractable counting of theory models and its application to truth maintenance and belief revision." *Journal of Applied Non-Classical Logics* 11, no. 1-2 (2001): 11-34.

We can also perform weighted model counting

$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

Suppose we have weights for variables

$\boldsymbol{X}$	$\mathbf{weight}(\boldsymbol{X})$	$\mathbf{weight}(\neg\boldsymbol{X})$
X_1	0.8	0.2
X_2	0.3	0.7
X_3	0.6	0.4

Let's look at the truth table first

X_1	X_2	X_3	α	weight
$\top$	$\top$	$\top$	$\bot$	
$\top$	$\top$	$\bot$	$\top$	$0.8 \times 0.3 \times 0.4 = 0.096$
$\top$	$\bot$	$\top$	$\top$	$0.8 \times 0.7 \times 0.6 = 0.336$
$\top$	$\bot$	$\bot$	$\bot$	
$\bot$	$\top$	$\top$	$\bot$	
$\bot$	$\top$	$\bot$	$\bot$	
$\bot$	$\bot$	$\top$	$\bot$	
$\bot$	$\bot$	$\bot$	$\bot$	

Total weight
$= 0.096 + 0.336$
$= 0.432$

Only these two rows matter because they are the only assignments that make the formula `true`

Darwiche, Adnan. "On the tractable counting of theory models and its application to truth maintenance and belief revision." *Journal of Applied Non-Classical Logics* 11, no. 1-2 (2001): 11-34.

We can also perform weighted model counting

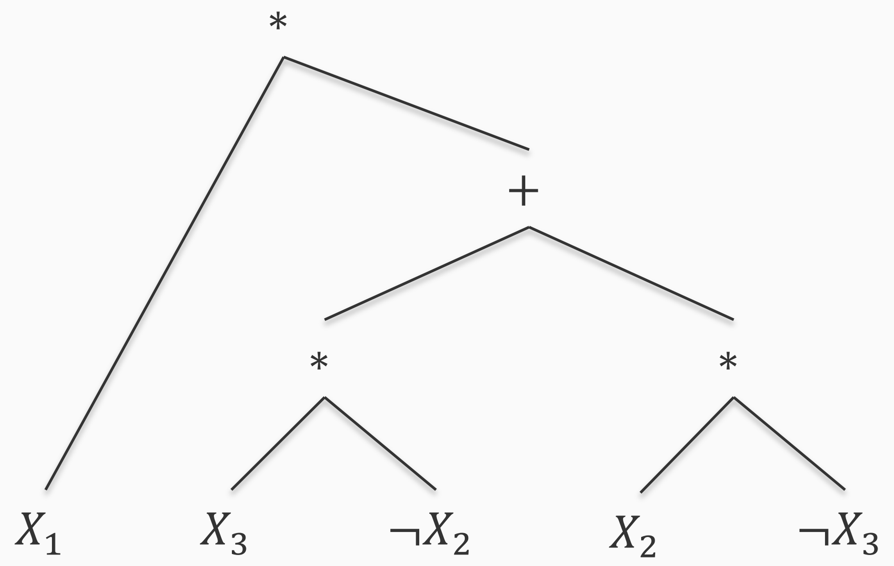
$$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3) \text{ is the same as } X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$$

Suppose we have weights for variables

$\boldsymbol{X}$	$\mathbf{weight}(\boldsymbol{X})$	$\mathbf{weight}(\neg\boldsymbol{X})$
X_1	0.8	0.2
X_2	0.3	0.7
X_3	0.6	0.4

Total weight of satisfying assignments $= 0.432$

We can compute this weight using the counting tree



Darwiche, Adnan. "On the tractable counting of theory models and its application to truth maintenance and belief revision." *Journal of Applied Non-Classical Logics* 11, no. 1-2 (2001): 11-34.

We can also perform weighted model counting

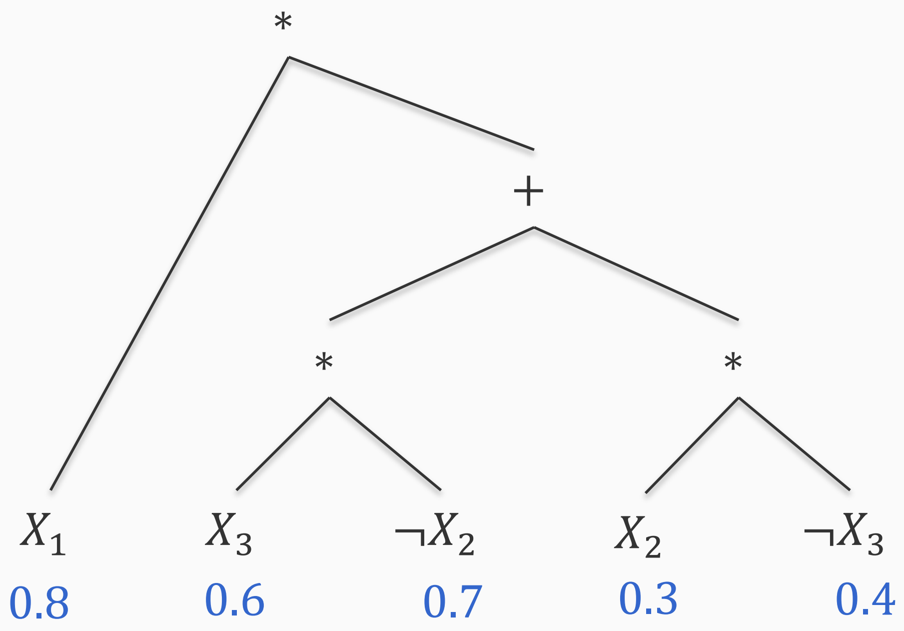
$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$

Suppose we have weights for variables

$\boldsymbol{X}$	$\mathbf{weight}(\boldsymbol{X})$	$\mathbf{weight}(\neg\boldsymbol{X})$
X_1	0.8	0.2
X_2	0.3	0.7
X_3	0.6	0.4

Total weight of satisfying assignments $= 0.432$

We can compute this weight using the counting tree



First, assign all the literals their respective weights

Darwiche, Adnan. "On the tractable counting of theory models and its application to truth maintenance and belief revision." *Journal of Applied Non-Classical Logics* 11, no. 1-2 (2001): 11-34.

We can also perform weighted model counting

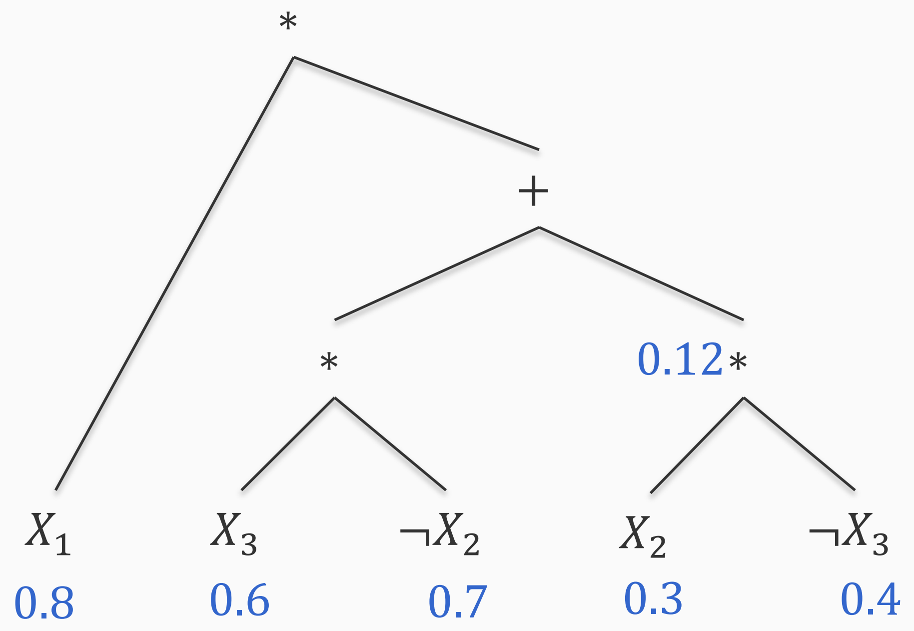
$\alpha = X_1 \land \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \land \big((X_2 \land \neg X_3) \lor (\neg X_2 \land X_3)\big)$

Suppose we have weights for variables

$\boldsymbol{X}$	$\mathbf{weight}(\boldsymbol{X})$	$\mathbf{weight}(\neg \boldsymbol{X})$
X_1	0.8	0.2
X_2	0.3	0.7
X_3	0.6	0.4

Total weight of satisfying assignments $= 0.432$

We can compute this weight using the counting tree



First, assign all the literals their respective weights

Next, propagate forward

Darwiche, Adnan. "On the tractable counting of theory models and its application to truth maintenance and belief revision." *Journal of Applied Non-Classical Logics* 11, no. 1-2 (2001): 11-34.

We can also perform weighted model counting

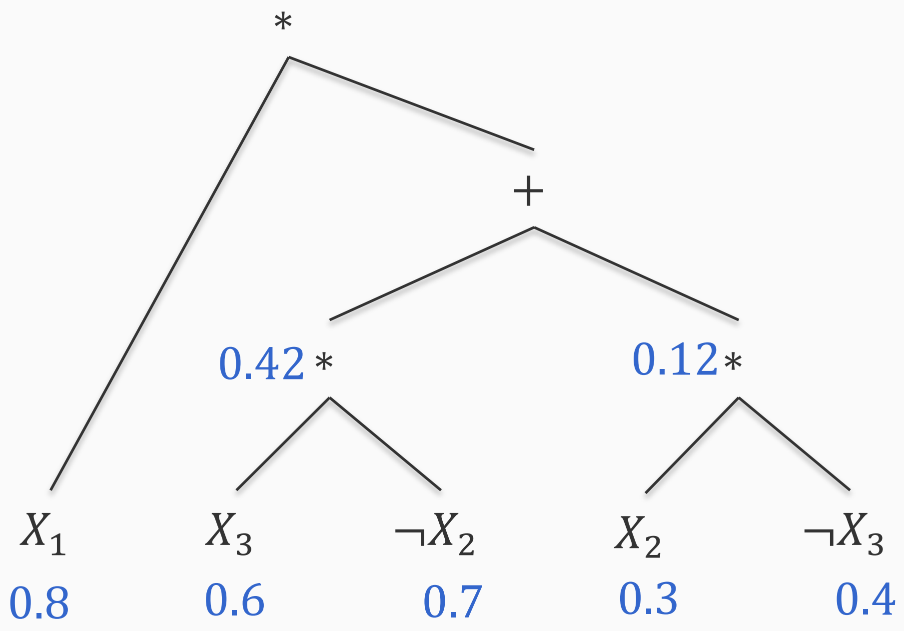
$$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3) \text{ is the same as } X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$$

Suppose we have weights for variables

$\boldsymbol{X}$	weight($\boldsymbol{X}$)	weight($\neg\boldsymbol{X}$)
X_1	0.8	0.2
X_2	0.3	0.7
X_3	0.6	0.4

Total weight of satisfying assignments $= 0.432$

We can compute this weight using the counting tree



First, assign all the literals their respective weights

Next, propagate forward

Darwiche, Adnan. "On the tractable counting of theory models and its application to truth maintenance and belief revision." *Journal of Applied Non-Classical Logics* 11, no. 1-2 (2001): 11-34.

We can also perform weighted model counting

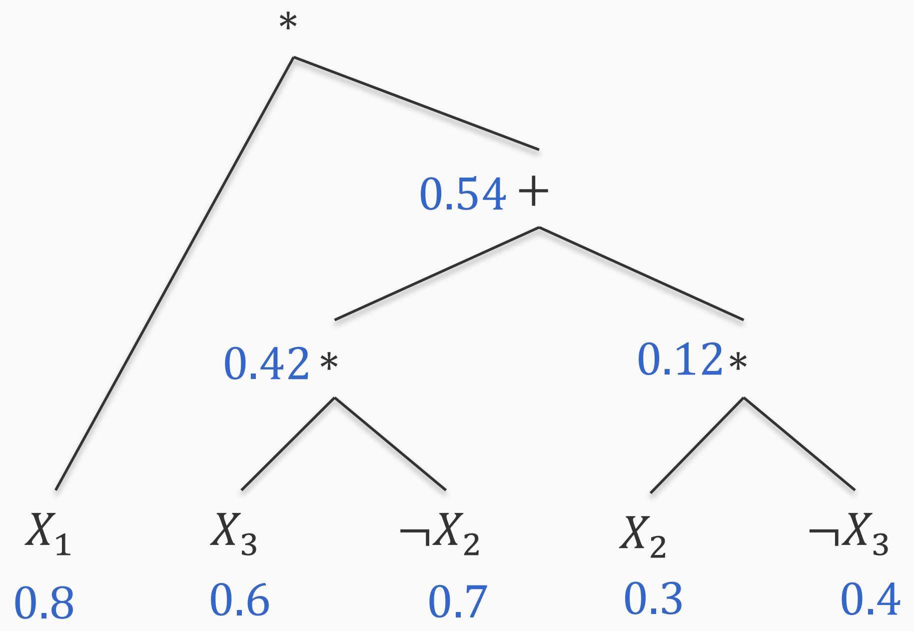
$\alpha = X_1 \land \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \land \big((X_2 \land \neg X_3) \lor (\neg X_2 \land X_3)\big)$

Suppose we have weights for variables

$\boldsymbol{X}$	$\mathbf{weight}(\boldsymbol{X})$	$\mathbf{weight}(\neg \boldsymbol{X})$
X_1	0.8	0.2
X_2	0.3	0.7
X_3	0.6	0.4

Total weight of satisfying assignments $= 0.432$

We can compute this weight using the counting tree



First, assign all the literals their respective weights

Next, propagate forward

Darwiche, Adnan. "On the tractable counting of theory models and its application to truth maintenance and belief revision." *Journal of Applied Non-Classical Logics* 11, no. 1-2 (2001): 11-34.

We can also perform weighted model counting

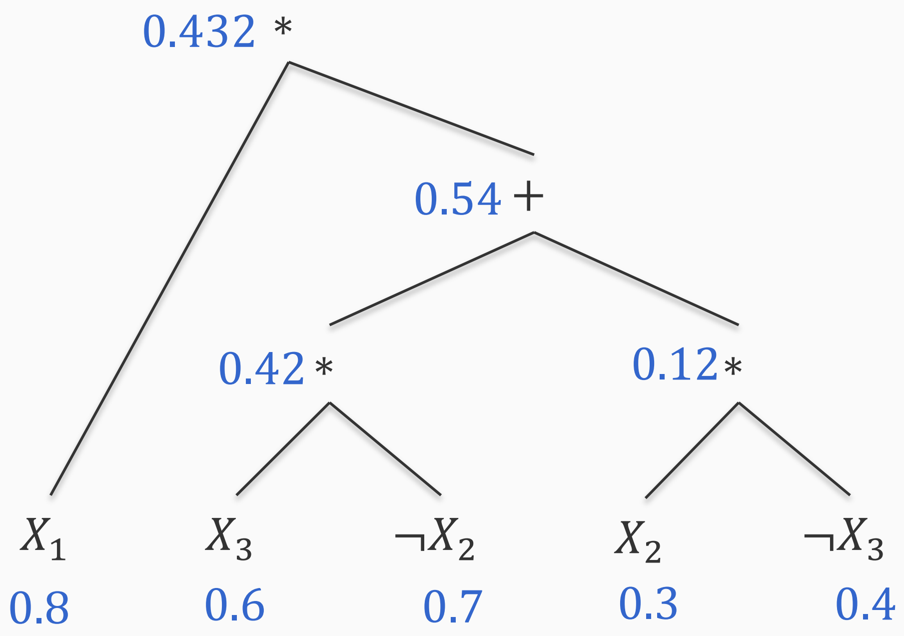
$$\alpha = X_1 \wedge \neg(X_2 \leftrightarrow X_3) \text{ is the same as } X_1 \wedge \big((X_2 \wedge \neg X_3) \vee (\neg X_2 \wedge X_3)\big)$$

Suppose we have weights for variables

$\boldsymbol{X}$	$\mathbf{weight}(\boldsymbol{X})$	$\mathbf{weight}(\neg\boldsymbol{X})$
X_1	0.8	0.2
X_2	0.3	0.7
X_3	0.6	0.4

Total weight of satisfying assignments $= 0.432$

We can compute this weight using the counting tree



First, assign all the literals their respective weights

Next, propagate forward

Darwiche, Adnan. "On the tractable counting of theory models and its application to truth maintenance and belief revision." *Journal of Applied Non-Classical Logics* 11, no. 1-2 (2001): 11-34.

We can also perform weighted model counting

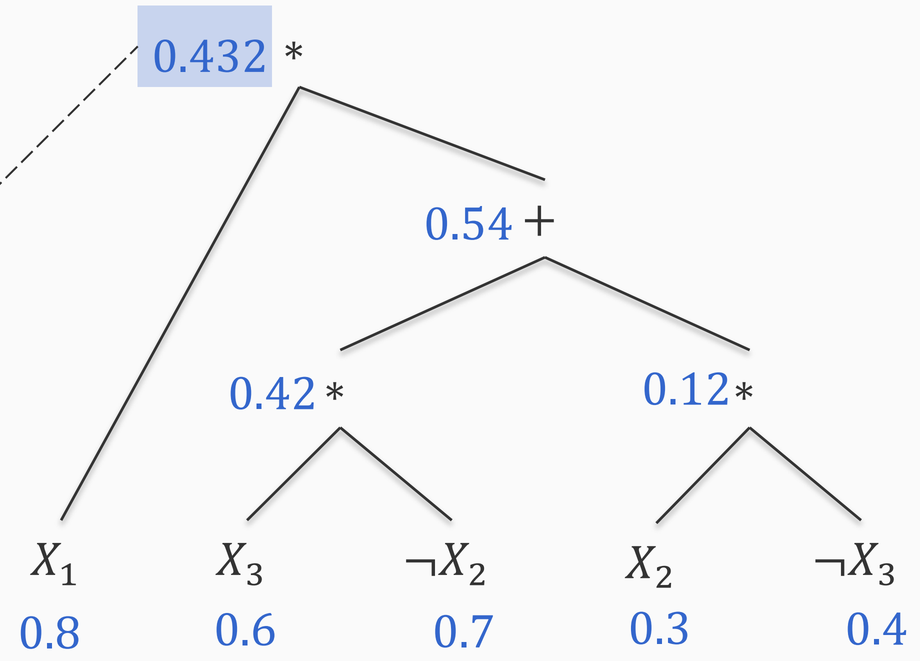
$\alpha = X_1 \land \neg(X_2 \leftrightarrow X_3)$ is the same as $X_1 \land \left((X_2 \land \neg X_3) \lor (\neg X_2 \land X_3)\right)$

Suppose we have weights for variables

$\boldsymbol{X}$	weight($\boldsymbol{X}$)	weight($\neg\boldsymbol{X}$)
X_1	0.8	0.2
X_2	0.3	0.7
X_3	0.6	0.4

Total weight of satisfying assignments $= 0.432$

We can compute this weight using the counting tree



First, assign all the literals their respective weights

Next, propagate forward

The value at the root is the total weighted model count!

Darwiche, Adnan. "On the tractable counting of theory models and its application to truth maintenance and belief revision." *Journal of Applied Non-Classical Logics* 11, no. 1-2 (2001): 11-34.

A second look at the semantic loss

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

This term requires us to accumulate quantities computed for every satisfying assignment for the formula α

Is this a problem? Have we seen this before?

Computing the semantic loss requires us to perform weighted model counting

Intractable in the worst case, *but tractable subsets of logic exist*

A second look at the semantic loss

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

This term requires us to accumulate quantities computed for every satisfying assignment for the formula α

Is this a problem? Have we seen this before?

Computing the semantic loss requires us to perform weighted model counting

Intractable in the worst case, *but tractable subsets of logic exist*

$$L(\alpha, \mathbf{p}) \propto -\log \boxed{WMC(\alpha, \mathbf{p})}$$

Knowledge compilation can help

Knowledge compilation: The process of converting a propositional knowledge base into a form that better supports certain kinds of queries

In our case, we can compile any propositional formula into a d-DNNF, which allows for two operations efficiently (if the resulting d-DNNF is not too large):

1. We can perform model counting and weighted model counting efficiently

2. We can take partial derivatives with respect to inputs efficiently

Logic as loss: Semantic loss

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

A simple semi-supervised learning example

Suppose we have:

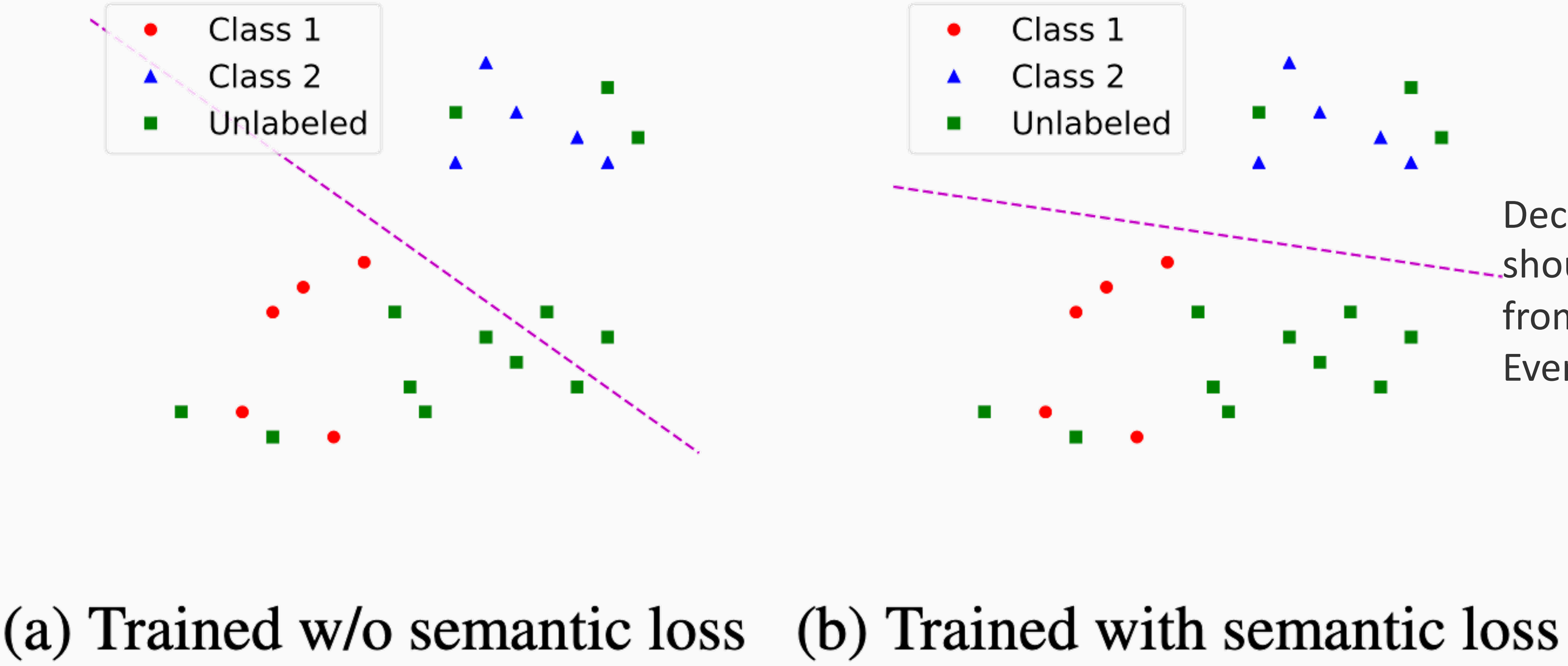
- A small number of labeled examples for a task with k labels
- A large collection of unlabeled examples

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?

A binary classification example



Decision boundaries should be far away from all examples. Even unlabeled ones

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*

The exactly-one constraint

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`

$$X_1 \vee X_2 \vee X_3$$

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

$$\neg X_1 \vee \neg X_2$$
$$\neg X_2 \vee \neg X_3$$
$$\neg X_3 \vee \neg X_1$$

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 original constraint

$$(X_1 \lor X_2 \lor X_3) \land (\neg X_1 \lor \neg X_2) \land (\neg X_2 \lor \neg X_3) \land (\neg X_3 \lor \neg X_1)$$

This is not in the deterministic decomposable negation normal form

The compiled exactly-one constraint

The original constraint

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

This is not in the deterministic decomposable negation normal form

But we can compile it to produce the following equivalent d-DNNF expression

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

Refer: The work by Adnan Darwiche in the 2000s that defines the normal form, analyzes complexity of querying it, and shows how to convert arbitrary Boolean formulas to the form

The compiled exactly-one constraint

The original constraint

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

This is not in the deterministic decomposable negation normal form

But we can compile it to produce the following equivalent d-DNNF expression

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

Refer: The work by Adnan Darwiche in the 2000s that defines the normal form, analyzes complexity of querying it, and shows how to convert arbitrary Boolean formulas to the form

Important: Because the semantic loss does not depend on the syntactic form that we use to define the constraint, we are free to use the more efficient form

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